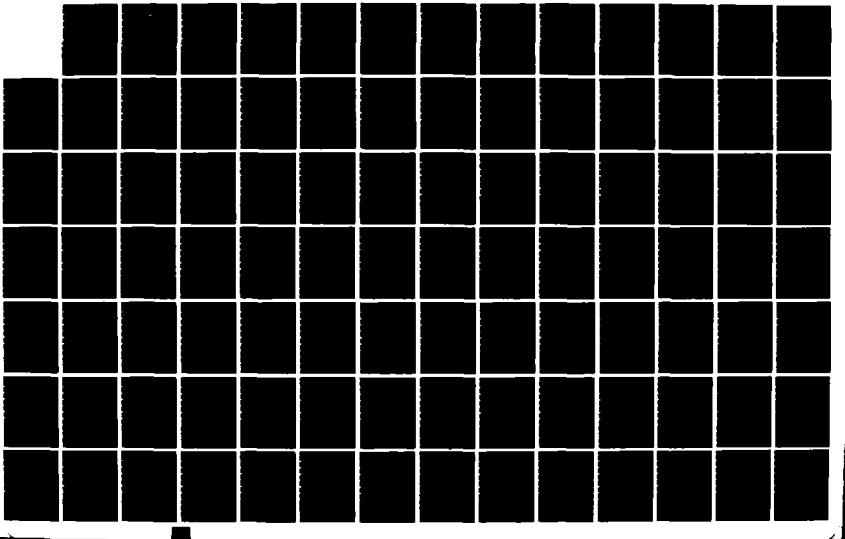
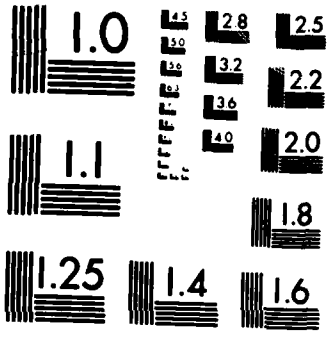
AD-A141 225 FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION(U) AIR FORCE INST OF TECH WRIGHT-PATTESSON AFB OH SCHOOL OF ENGI.. A D GOFF 1/2

UNCLASSIFIED MAR 84 AFIT/GNE/PH/84M-4 F/G 12/1 NL





FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION

THESIS

Allan D. Goff
First Lieutenant, USAF

AFIT/GNE/PH/84M-4

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

FINITE ELEMENT SOLUTION Of A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION

THESIS

Allan D. Goff
First Lieutenant, USAF

AFIT/GNE/PH/84M-4

AFIT/GNE/PH/84M-4

FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirement for the Degree of
Master of Science in Nuclear Engineering

Allan D. Goff, B. S.
First Lieutenant, USAF

March 1984

Preface

The purpose of this study was to explore the feasibility of using the finite element method in a numerical solution of the transport equation. For problems in science and engineering, the finite element method can offer significant advantages over competing numerical techniques, such as the method of finite difference. In general, it requires that the differential equation modeling the physical system be mathematically self-adjoint. The finite element method has not been used in the past on the transport equation for precisely this reason: it is not, in its current formulation, a self-adjoint equation. In this study, the transport equation was recast into a form which, while more complex, is indeed self-adjoint. The finite element equations were derived for the one-dimensional case with isotropic scatter. This proved to be one of the challenges in this study due to the non-local character of the transport equation. The term "non-local" means that the solution at a given point is dependent upon points which, in phase space, are far removed from the point of interest. Lastly, a numerical solution was written in FORTRAN and implemented on a VAX 11/780. The results were compared against a benchmark solution using a recently developed solution technique called Ln.

As in all research efforts, several people played important roles. I would like to particularly thank Dr. Donn Shankland for the technical guidance he provided. His contribution to this research effort was significant. I would also like to express my gratitude to Dr. John Jones for his suggestions on the finite element method. Finally, I would like to thank my wife, Jamie, whose love and understanding endured through this graduate program.

Contents

List of Figures

List of Tables

Notation

Symbol	Definition
Constants:	
π	Pi
λ	Signed unity constant, negative for labeled cases
Coordinates:	
$\hat{e}_x, \hat{e}_y, \hat{e}_z$	Unit vectors in cartesian coordinates
ϕ	Azimuthal angle in spherical coordinates
θ	Polar angle in spherical coordinates
x	X-axis in the $x\mu$-plane
μ	μ-axis in the $x\mu$-plane
y	Y-axis in the $y\mu'$-plane
μ'	μ-axis in the $y\mu'$-plane
Operators:	
d	Derivative
∂	Partial derivative
∇	Gradient
δ	First variation
$\mathcal{L}$	Operator kernal in transport equation
L	Self-Adjoint Operator
L	Operator kernal in self-adjoint transport equation
$\dagger$	Adjoint
$*$	Complex conjugate

$\sim$ - Transpose

$| \; |$ - Absolute magnitude

$\sqrt{\;}$ - Square root

$\sin$ - Sine function

$\cos$ - Cosine function

Integrals:

$\int$ - Integral sign

I - Extremization integral

I_1 - Streaming term in extremization integral

I_2 - Absorption term in extremization integral

I_3 - First scatter term in extremization integral

I_4 - Second scatter term in extremization integral

I_5 - Third scatter term in extremization integral

$\int_D dx$ - Integration over the entire domain

d^3r - Integration over all the space variables

d^3v - Integration over all the velocity variables

$\vec{dS}$ - Integration over the surface bounding some volume

dx - Integration over the x-axis

$d\mu$ - Integration over the μ-axis

$d\mu'$ - Integration over the μ'-axis

dA - Integration over an entire element

$\bar{\mu}'$ - Upper Limit of integration on non-local element

$\underline{\mu}'$ - Lower limit of integration on non-local element

Σ - Summation sign

Vectors:

$\vec{r}$ - Position vector

$\vec{v}$ - Velocity vector

$\vec{v}'$ - Pre-collision velocity vector

$\underline{\mathcal{Q}}$ - Vector of interior nodes

$\underline{\Psi}$ - Vector of boundary nodes

$\underline{x}$ - Vector of the x-coordinates of the nodes of some element

$\underline{\mu}$ - Vector of the μ-coordinates of the nodes of some element

$\underline{\mu}'$ - Vector of the μ'-coordinates of the nodes of some element

$\underline{I}$ - Unit vector

$\underline{L}$ - Vector of triangular coordinates

Matrices:

$\underline{\underline{M}}$ - Global matrix

$\underline{\underline{M}}_{11}$ - Global matrix, interior nodes to interior nodes

$\underline{\underline{M}}_{12}$ - Global matrix, interior nodes to boundary nodes

$\underline{\underline{M}}_{21}$ - Global matrix, boundary nodes to interior nodes

$\underline{\underline{M}}_{22}$ - Global matrix, boundary nodes to boundary nodes

m_{ij} - Local matrix

$m_{ij}^{(1)}$ - Local matrix from streaming term

$m_{ij}^{(a)}$ - Local matrix from absorption term

$m_{ij}^{(s)}$ - Local matrix from first scatter term

$m_{ij}^{(s)}$ - Local matrix from second scatter term

$m_{ij}^{(s)}$ - Local matrix from third scatter term

$\underline{\underline{m}}_l$ - Integer matrix in scatter terms from x_l

$\underline{\underline{m}}_2$ - Integer matrix in scatter terms from x_2

$\underline{\underline{m}}_3$ - Integer matrix in scatter terms from x_3

$\underline{\underline{m}}_4$ - Integer matrix in scatter terms equal to sum of above

$\underline{\underline{m}}_5$ - Integer matrix in scatter terms from x_1 and x_2

$\underline{\underline{m}}_6$ - Integer matrix first scatter term, symmetric cases

$\underline{\underline{m}}_7$ - Integer matrix first scatter term, antisymmetric cases

Transport equation:

t - Time

v - Velocity

V - Volume

S - Surface area

n - Number of particles

$N(\vec{r},t)$ - Number density, configuration space

$n(\vec{r},\vec{v},t)$ - Number density, phase space

mfp - Mean free path

Σ - Macroscopic cross-section

Σ_t - Macroscopic cross-section, total

Σ_s - Macroscopic cross-section, scatter

$\Sigma(\vec{r},\vec{v}'\rightarrow\vec{v})$ - Scattering kernal

$f(\vec{r},\vec{v}'\rightarrow\vec{v})$ - Scattering probability function

$c(\vec{r},\vec{v}')$ - Mean number of secondary particles emitted in a collision

$\Phi(\vec{r},t)$ - Scalar flux

$\varphi(\vec{r},\vec{v},t)$ - Phase space scalar flux

$\vec{J}(\vec{r},\vec{v},t)$ - Phase space current density

$S(\vec{r},\vec{v},t)$ - Phase space source density

$\hat{\Omega}$ - Direction cosine in 3-D, local

$\hat{\Omega}'$ - Direction cosine in 3-D, non-local

μ - Direction cosine in 1-D, local

μ' - Direction cosine in 1-D, non-local

ρ - Effective source in the self-adjoint transport equation

α - Constant in first scatter term in extremization integral

Finite elements:

x_i - x-coordinate of the local elements' nodes

μ_i - μ-coordinate of the local elements' nodes

y_i - x-coordinate of the non-local elements' nodes

μ_i' - μ-coordinate of the non-local elements' nodes

ϕ_i - Nodal values

N_i - Linear interpolating functions, local

a_i - x-coefficient in linear interpolating functions, local

b_i - μ-coefficient in linear interpolating functions, local

c_i - Constant in the linear interpolating functions, local

N_j' - Linear interpolating functions, non-local

a_j' - x-coefficient in linear interpolating functions, non-local

b_j' - μ'-coefficient in linear interpolating functions, non-local

c_j' - Constant in the linear interpolating functions, non-local

A - Area of the local element

A' - Area of the non-local element

L_i - Triangular coordinates, local

L'_j - Triangular coordinates, non-local

p, q, r - Exponents in the triangular integration formula

ℓ - Collected u' terms: $u'_3 + u'_2 - 2u'_1$

k - Collected u' terms: $u'_3 + u'_2 - 2u'_1 \frac{y_1}{y_2}$

c - Collected u' terms: $\frac{1}{2}(\ell + k)$

d - Width of column

P - Constant factor in local matrices

Abstract

A self-adjoint form of the transport equation was derived, and expressed as an extremization integral. The finite element equations were derived from the extremization integral for the one-dimensional time independent homogeneous transport equation with isotropic scatter. These equations were implemented in FORTRAN on a VAX 11/780 and used to solve a simple benchmark problem.

The finite element solution, for a small mesh of 32 elements, was compared to results from a numerical technique known as Ln. The solutions differed by about 35 percent. Larger meshes were not run because an automatic mesh refinement routine was not available. The large difference between the Ln solution and the finite difference solution is attributed to residual errors in the coding of the finite element equations for the scatter terms.

FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION

I. Introduction

Background

The transport equation is a remarkably successful description of transport phenomena. It has been used on a wide range of problems in such diverse fields as nuclear reactors, astrophysics, charged particle transport, rarefied gas dynamics, and even traffic flow.

A variety of solution techniques have been applied to the transport equation (4:2), which may be roughly divided into four classes: analytic, approximation, numerical, and simulation. Except for the very simplest problems, the analytic solutions tend to be specialized to some particular field, or problem, and therefore lack generality. The standard approximation schemes are complicated by the non-self-adjoint nature of the transport equation, and they often require considerable modification. Simulations, such as Monte Carlo, are capable of very high accuracy, but the cost is often prohibitive due to the large number of computer operations required.

Numerical techniques offer a good engineering compromise. They are reasonably accurate at a reasonable cost and

are generally applicable over a wide range of problems.

The finite element method is a relatively recent (7:vii) numerical technique which has been successfully applied in many fields. It has been applied to the transport equation, without notable success, due to the non-self-adjoint nature of the transport equation (4:479-504).

Objective

The objective of this study was to recast the transport equation into a form which is self-adjoint, and then apply the finite element method toward its solution.

Scope

The transport equation was recast into a self-adjoint form and expressed as an extremization principle. The finite element equations were derived from this extremization principle for the homogeneous transport equation with isotropic scatter. They were then implemented in FORTRAN on a VAX 11/780 for a simple benchmark problem. Two sets of computer runs were done. In the first set, the results for the transport equation without scatter were compared to an analytic solution. In the second set, the results for the transport equation with scatter were compared to the results from another numerical technique called Ln.

Assumptions

The assumptions in this study are those commonly used

in deriving the transport equation, those used in making the finite element approximation, and those used in defining the benchmark problem. They are described in the relevant sections of this thesis.

Sequence of Presentation

The next chapter reviews the transport equation and presents the derivation of the self-adjoint form. In Chapter III, the finite element equations are derived. The computer implementation is described in Chapter IV. Chapter V presents the results of the computer program and Chapter VI gives the conclusions and recommendations.

II. The Self-Adjoint Transport Equation

The Transport Equation

The term "transport theory" is commonly used to refer to the mathematical description of the motion of particles through a host medium. It differs from the usual approaches used in classical physics because it is a particle, rather than a continuum, theory of matter. The concept of a continuous field is, however, employed in transport theory in probability fields for some of the properties of the system of particles.

There are several common assumptions in transport theory. In this development, the following assumptions were made:

The particles interact only with the host medium.

There is no correlation between particles.

Only the particle's position and velocity are significant.

All interactions with the host medium occur at a point well localized in space with respect to the mean free path and well localized in time with respect to the mean time of flight.

Transport theory rests on several key concepts which are reviewed briefly here. Greater detail can be found in Transport Theory, by Duderstadt and Martin (4:1-19).

The first and most central concept is that of a probability field which describes the expected density of

particles in configuration space. Specifically, the particle density is defined by

$$N(\vec{r},t)\,d^3r \equiv \text{expected number of particles in } d^3r \text{ about } \vec{r} \text{ at time t.}$$

Similarly, the phase space particle density is defined by

$$n(\vec{r},\vec{v},t)\,d^3r\,d^3v \equiv \text{expected number of particles in } d^3r \text{ about } \vec{r} \text{ with velocity in } d^3v \text{ at time t.}$$

The two are related by

$$N(\vec{r},t) = \int n(\vec{r},\vec{v},t)\,d^3v \qquad (1)$$

The second key concept is the representation of interactions between the transporting particles and the host medium. Two terms are used in this concept. They are "mean free path" (mfp) and "macroscopic cross-section." They are inverses of each other and are defined by

$$(mfp)^{-1} = \Sigma(\vec{r},\vec{v}) \equiv \text{probability of particle interaction per unit distance traveled by a particle of velocity } \vec{v} \text{ at position } \vec{r}.$$

As a further refinement, the collision kernel is defined as the

$$\Sigma(\vec{r},\vec{v}'\to\vec{v}) \equiv \text{probability per unit distance traveled that an incident particle of velocity } \vec{v}' \text{ will collide producing a secondary particle at velocity } \vec{v}.$$

Mathematically

$$\Sigma(\vec{r}, \vec{v}' \rightarrow \vec{v}) \equiv \Sigma(\vec{r}, \vec{v}')\, c(\vec{r}, \vec{v}')\, f(\vec{r}, \vec{v}' \rightarrow \vec{v}) \tag{2}$$

where c is the mean number of secondary particles emitted per collision event and is defined by

$c(\vec{r}, \vec{v}) \equiv$ mean number of secondary particles emitted in a collision event experienced by an incident particle with velocity $\vec{v}$ at position $\vec{r}$.

and where f is the scattering probability function defined by

$f(\vec{r}, \vec{v} \rightarrow \vec{v})\, d^3v \equiv$ probability that any secondary particles induced by an incident particle with velocity $\vec{v}$ will be emitted with velocity $\vec{v}$ in d^3v.

Note that

$$\int f(\vec{r}, \vec{v}' \rightarrow \vec{v})\, d^3v = 1 \tag{3}$$

When the macroscopic cross-section is independent of position and velocity it is symbolized by Σ_t for total.

In addition to these key concepts, there are a few supporting concepts. The particle scalar flux is defined by

$$\Phi(\vec{r}, t) \equiv \int \phi(\vec{r}, \vec{v}, t)\, d^3v \tag{4}$$

where $\phi(\vec{r}, \vec{v}, t)$ is the particle scalar phase space flux and is defined by

$$\phi(\vec{r}, \vec{v}, t) \equiv v\, n(\vec{r}, \vec{v}, t) \tag{5}$$

The term "flux" is usually used by itself, with the adjectives dropped, relying on context to identify the desired meaning. This use of the term is, except in the nuclear community, non-standard and should not be confused with the vector flux. The phase space current density is defined as

$$\vec{J}(\vec{r},\vec{v},t) \equiv \vec{v}\, n(\vec{r},\vec{v},t) \tag{6}$$

where

$\vec{J}(\vec{r},\vec{v},t)\cdot d\vec{S}\, d^3v \equiv$ expected number of particles that cross an area dS per second with velocity $\vec{v}$ in d^3v at time t.

The last supporting concept is of a unit vector in the direction of the velocity vector, called the cosine vector. It is defined by

$$\hat{\Omega} \equiv \frac{\vec{v}}{|\vec{v}|} = \cos\phi \sin\theta \hat{e}_x + \sin\phi \sin\theta \hat{e}_y + \cos\theta \hat{e}_z \tag{7}$$

where θ and ϕ have their usual meanings in spherical coordinates and $\hat{e}_x$, $\hat{e}_y$ and $\hat{e}_z$ are the unit vectors in cartesian coordinates.

The transport equation can be derived by invoking a balance condition around the conservation of particles. In an arbitrarily small, fixed region in space with volume V, surface area S and number of particles n, the balance condition is

$$\left\{\begin{matrix}\text{time rate}\\ \text{of change}\\ \text{of n}\end{matrix}\right\} = \left\{\begin{matrix}\text{change due}\\ \text{to leakage}\\ \text{thru S}\end{matrix}\right\} + \left\{\begin{matrix}\text{change}\\ \text{due to}\\ \text{collisions}\end{matrix}\right\} + \left\{\text{sources}\right\}$$

Writing the balance condition mathematically gives

$$\frac{\partial}{\partial t}\int_V \left[n(\vec{r},\vec{v},t)\,d^3v\right] d^3r = -\int_S \left[\vec{J}(\vec{r},\vec{v},t)\,d^3v\right]\cdot d\vec{S} + \int_V \left[\left(\frac{\partial n}{\partial t}\right)_{coll} d^3v\right] d^3r + \int_V \left[s(\vec{r},\vec{v},t)\,d^3v\right] d^3r \quad (8)$$

The leakage term can be rewritten using Gauss's law and using the independence between the space and velocity variables as

$$\int_S \left[\vec{J}(\vec{r},\vec{v},t)\,d^3v\right]\cdot d\vec{S} = \int_V \left[\nabla\cdot\vec{J}(\vec{r},\vec{v},t)\,d^3v\right] d^3r \quad (9)$$

$$\int_V \left[\nabla\cdot\vec{J}(\vec{r},\vec{v},t)\,d^3v\right] d^3r = \int_V \left[\vec{v}\cdot\nabla n(\vec{r},\vec{v},t)\,d^3v\right] d^3r \quad (10)$$

Thus, the balance condition becomes

$$\int\left\{\left[\frac{\partial n}{\partial t} + \vec{v}\cdot\nabla n - \left(\frac{\partial n}{\partial t}\right)_{coll} - S\right] d^3v\right\} d^3r = 0 \quad (11)$$

Since the volume is arbitrary

$$\frac{\partial n}{\partial t} + \vec{v}\cdot\nabla n - \left(\frac{\partial n}{\partial t}\right)_{coll} = S \quad (12)$$

The collision term can be represented more precisely by

$$\left(\frac{\partial n}{\partial t}\right)_{coll} = \int v'\Sigma(\vec{r},\vec{v}'\to\vec{v})\,n(\vec{r},\vec{v}',t)\,d^3v' - v\Sigma(\vec{r},\vec{v})\,n(\vec{r},\vec{v},t) \quad (13)$$

which, under the assumption of uniform host medium, becomes

$$\left(\frac{\partial n}{\partial t}\right)_{coll} = \int v'\Sigma(\vec{v}'\to\vec{v})\,n(\vec{r},\vec{v}',t)\,d^3v' - v\Sigma_t(\vec{v})\,n(\vec{r},\vec{v},t) \quad (14)$$

Substitution gives

$$\frac{\partial n}{\partial t} + \vec{v}\cdot\nabla n + v\Sigma_t(\vec{v})n = \int\Sigma(\vec{v}'\to\vec{v})\,v'n(\vec{r},\vec{v}',t)\,d^3v' + S \quad (15)$$

Rewriting it in terms of flux and cosine vectors, gives

$$\frac{1}{v}\frac{\partial \phi}{\partial t} + \hat{\Omega}\cdot\nabla\phi + \Sigma_t(\vec{r})\phi = \int_0^\infty dE' \int_{4\pi} \Sigma(E'\to E, \hat{\Omega}'\to\hat{\Omega})\,\phi(\vec{r}, E, \hat{\Omega}, t)\,d\hat{\Omega}' + S \quad (16)$$

While this form of the transport equation is very explicit, it is unwieldy for many mathematical manipulations. As a notational aid, an operator is defined such that the transport equation becomes

$$\mathcal{L}\phi - S = 0 \quad (17)$$

where, in this instance, the operator is

$$\mathcal{L} = \frac{1}{v}\frac{\partial}{\partial t} + \hat{\Omega}\cdot\nabla + \Sigma_t(\vec{r}) - \int_0^\infty dE' \int_{4\pi} \Sigma(E'\to E, \hat{\Omega}'\to\hat{\Omega})\,d\hat{\Omega}' \quad (18)$$

As in all differential equations, this statement of the problem is incomplete until boundary conditions are specified.

Depending on the application, several additional assumptions are commonly applied to simplify the mathematics of the transport equation. In this study, four additional assumptions were made.

The first assumption was that all sources are monoenergetic and that collision events result in either absorption or in elastic scatter. Thus, the transport equation becomes

$$\frac{1}{v}\frac{\partial \phi}{\partial t} + \hat{\Omega}\cdot\nabla\phi + \Sigma_t\phi - \int_{4\pi} \Sigma(\hat{\Omega}'\to\hat{\Omega})\,\phi(\vec{r}, \hat{\Omega}, t)\,d\hat{\Omega} = S \quad (19)$$

and is called the one speed model.

The second assumption was to ignore the time dependency, that is, only the steady state-solution is of interest. Thus, the transport equation becomes

$$\hat{\Omega}\cdot\nabla\phi + \Sigma_t\phi - \int_{4\pi}\Sigma(\hat{\Omega}'\rightarrow\hat{\Omega})\,\phi(\vec{r},\hat{\Omega})\,d\hat{\Omega} = S \qquad (20)$$

The third assumption was that the region of interest had planar geometry, i.e., that it could be modeled in one dimension. Aligning the coordinate system with the z-axis perpendicular to the plane of symmetry simplifies the divergence to

$$\nabla\phi = \frac{\partial\phi}{\partial z}\,\hat{e}_z \qquad (21)$$

Therefore

$$\hat{\Omega}\cdot\nabla\phi = \cos\theta\,\frac{\partial\phi}{\partial z} \qquad (22)$$

letting

$$\mu = \cos\theta \qquad (23)$$

and changing the label on the z-axis to x yields

$$\hat{\Omega}\cdot\nabla\phi = \mu\frac{\partial\phi}{\partial x} \qquad (24)$$

Additionally, expanding the collision kernel yields

$$\Sigma(\hat{\Omega}'\rightarrow\hat{\Omega}) = c\,\Sigma_t f(\hat{\Omega}'\rightarrow\hat{\Omega}) = \Sigma_s f(\hat{\Omega}'\rightarrow\hat{\Omega}) \qquad (25)$$

so the one-dimensional transport equation becomes

$$\mu\frac{\partial\phi}{\partial x} + \Sigma_t\phi - \Sigma_s\int f(\hat{\Omega}'\rightarrow\hat{\Omega})\,\phi(x,\mu')\,d\hat{\Omega}' = S \qquad (26)$$

The fourth and final assumption was that of isotropic scatter. Recall that

$$\int f(\vec{\Omega}' \to \vec{\Omega})\, d^2\vec{\Omega}' = 1 \tag{27}$$

since

$$d\vec{\Omega} = \sin\theta\, d\theta\, d\phi = -du\, d\phi \tag{28}$$

This becomes

$$\int_0^{2\pi}\int_0^{\pi} f(\vec{\Omega}' \to \vec{\Omega}) \sin\theta'\, d\theta'\, d\phi' = 1 \tag{29}$$

Since, for isotropic scatter, $f(\vec{\Omega}' \to \vec{\Omega})$ is a constant

$$\int_0^{2\pi}\int_{-1}^{+1} f\, du'\, d\phi' = 1 \tag{30}$$

which implies

$$f(\vec{\Omega}' \to \vec{\Omega}) = \frac{1}{4\pi} \tag{31}$$

Therefore,

$$\Sigma_s \int f(\vec{\Omega}' \to \vec{\Omega})\, \phi(x,\mu')\, d\vec{\Omega}' = \frac{\Sigma_s}{4\pi}\int_0^{2\pi} d\phi' \int_{-1}^{+1} \phi(x,\mu')\, du' \tag{32}$$

or

$$\Sigma_s \int f(\vec{\Omega}' \to \vec{\Omega})\, \phi(x,\mu')\, d\vec{\Omega}' = \frac{\Sigma_s}{2}\int_{-1}^{+1} \phi(x,\mu')\, du' \tag{33}$$

Thus, the one-speed, time independent, one-dimensional transport equation in a uniform and isotropic medium with isotropic scatter becomes

$$\mu\frac{\partial\phi}{\partial x} + \Sigma_t\phi - \frac{\Sigma_s}{2}\int_{-1}^{+1}\phi(x,\mu')\, du' = S \tag{34}$$

The operator form is the same for all of these simplified transport equations

$$\mathcal{L}\phi - S = 0 \tag{35}$$

where, for the one-dimensional case with isotropic scatter, the integro-differential operator is

$$\mathcal{L} = \mu \frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2} \int_{-1}^{+1} d\mu' \tag{36}$$

Mathematical Background

Continuum problems often have different, but equivalent, formulations. In the differential formulation, the problem is to integrate an integro-differential equation subject to given boundary conditions. In the classical variational formulation, the problem is to find an unknown function which extremizes a functional subject to the same given boundary conditions. A familiar example of this dual formulation is the equivalence between Newtonian mechanics, a differential formulation, and Hamiltonian mechanics, a variational formulation (5:85-105).

There are some standard techniques used to translate a differential formulation into a variational formulation. One is to select a quadratic functional derived from the differential formulation. The usual choice for a quadratic (6:10) is

$$(Lv, v) - 2(f, v) \tag{37}$$

where the differential formulation was

$$L\nu - f = 0 \tag{38}$$

and with the definition that

$$(x, y) \equiv \int_D y^* x \, d\underline{x} \tag{39}$$

where $d\underline{x}$ signifies integration over the entire domain, and y* is the complex conjugate of y. Since all the variables in the transport equation are real valued, the complex conjugate notation will be suppressed. The integral to be extremized then becomes

$$I = \tfrac{1}{2}\left[\int_D \nu L \nu \, d\underline{x} - 2\int_D \nu f \, d\underline{x}\right] \tag{40}$$

Another, but seldom used, quadratic is

$$(L\nu - f,\ L\nu - f) \tag{41}$$

There is one restriction on the above approach -- the operator in the resulting variational formulation must be self-adjoint. Adjointness is defined (8:10) by the property

$$\int_D \phi \mathcal{L} \psi \, d\underline{x} = \int_D \psi \mathcal{L}^+ \phi \, d\underline{x} \tag{42}$$

where $\mathcal{L}^+$ is the adjoint of the operator $\mathcal{L}$.

A proof of this requirement follows. The transport equation is

$$\mathcal{L}\phi - S = 0 \tag{43}$$

and the standard quadratic is

$$(\mathcal{L}\phi, \phi) - 2(s, \phi) \tag{44}$$

so the integral to be extremized is

$$I = \tfrac{1}{2}\int_D \phi(\mathcal{L}\phi - 2s)\, d\underline{x} \tag{45}$$

Taking the variation

$$\delta I = \tfrac{1}{2}\delta\int_D \phi(\mathcal{L}\phi - 2s)\, d\underline{x} = 0 \tag{46}$$

and

$$\delta I = \tfrac{1}{2}\int_D \left\{\delta\phi(\mathcal{L}\phi - 2s) + \phi\mathcal{L}\delta\phi\right\} d\underline{x} = 0 \tag{47}$$

using the definition of adjointness

$$\delta I = \tfrac{1}{2}\int_D \left\{\delta\phi(\mathcal{L}\phi - 2s) + \delta\phi(\mathcal{L}^+\phi)\right\} d\underline{x} = 0 \tag{48}$$

Rearranging and combining terms

$$\delta I = \tfrac{1}{2}\int_D \delta\phi\left\{(\mathcal{L}+\mathcal{L}^+)\phi - 2s\right\} d\underline{x} = 0 \tag{49}$$

or

$$\delta I = \int_D \delta\phi\left\{\tfrac{1}{2}(\mathcal{L}+\mathcal{L}^+)\phi - s\right\} d\underline{x} = 0 \tag{50}$$

but since the variation is arbitrary

$$\tfrac{1}{2}(\mathcal{L}+\mathcal{L}^+)\phi - s = 0 \tag{51}$$

Clearly, this is equal to the transport equation only if $\mathcal{L}$ is self-adjoint. But $\mathcal{L}$ is not self-adjoint for the

transport equation, so something else must be tried.

Operator Formulation

A variational form of the transport equation can be derived by starting with a different quadratic. Specifically,

$$(\mathcal{L}\phi - S,\ \mathcal{L}\phi - S) \tag{52}$$

Now the integral to be extremized becomes

$$I = \tfrac{1}{2}\int_D \{(\mathcal{L}\phi - S)(\mathcal{L}\phi - S)\}\, d\mu \tag{53}$$

and

$$I = \tfrac{1}{2}\int_D \{\mathcal{L}\phi\mathcal{L}\phi - (\mathcal{L}\phi)S - S(\mathcal{L}\phi) + S^2\}\, d\mu \tag{54}$$

or

$$I = \tfrac{1}{2}\int_D \{\mathcal{L}\phi\mathcal{L}\phi - 2S\mathcal{L}\phi + S^2\}\, d\mu \tag{55}$$

Using the definition of adjointness

$$I = \tfrac{1}{2}\int_D \{\phi\mathcal{L}^+\mathcal{L}\phi - 2\phi\mathcal{L}^+S + S^2\}\, d\mu \tag{56}$$

Taking the variation

$$\delta I = \tfrac{1}{2}\delta\int_D \{\phi\mathcal{L}^+\mathcal{L}\phi - 2\phi\mathcal{L}^+S + S^2\}\, d\mu = 0 \tag{57}$$

and

$$\delta I = \tfrac{1}{2}\int_D \{\delta\phi(\mathcal{L}^+\mathcal{L}\phi - 2\mathcal{L}^+S) + \phi\mathcal{L}^+\mathcal{L}\delta\phi\}\, d\mu = 0 \tag{58}$$

again using the definition of adjointness and combining terms

yields

$$\delta I = \int_D \delta\phi \left(\mathcal{L}^+\mathcal{L}\phi - \mathcal{L}^+ s\right) d\mathcal{V} = 0 \tag{59}$$

or

$$\delta I = \int_D \delta\phi \left\{\mathcal{L}^+\left(\mathcal{L}\phi - s\right)\right\} d\mathcal{V} = 0 \tag{60}$$

Since the variation is arbitrary

$$\mathcal{L}^+\left(\mathcal{L}\phi - s\right) = 0 \tag{61}$$

Let

$$\psi = \mathcal{L}\phi - s \tag{62}$$

then there are clearly two solutions. Either

$$\mathcal{L}\phi - s = 0 \tag{63}$$

which is the solution to the transport equation or

$$\mathcal{L}^+\psi = 0 \tag{64}$$

which is not.

Equation (64) is just the homogeneous adjoint transport equation, which is equivalent to the transport equation without sources and with the velocity vectors reversed. Clearly, if there is any net absorption, the steady state solution is identically zero. Thus, this equation, applied to a region where there is net absorption, is zero because ψ is zero. Therefore, the only solution to the self-adjoint transport

equation is also the solution to the transport equation.

As a notational aid, the following definitions are made.

$$L = \mathcal{L}^{+}\mathcal{L} \tag{65}$$

and

$$\rho = \mathcal{L}^{+}S \tag{66}$$

Then the relation which specifies the extremization of the functional becomes

$$L\phi - \rho = 0 \tag{67}$$

which will henceforth be referred to as the self-adjoint transport equation.

Integral-Differential Formulation

From the operator form of the self-adjoint transport equation, it is a straightforward matter to derive the equation in its differential form. The operator, in the one-dimensional transport equation for the case of isotropic scatter is

$$\mathcal{L} = \mu\frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2}\int_{-1}^{+1} d\mu' \tag{68}$$

and its adjoint is

$$\mathcal{L}^{+} = -\mu\frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2}\int_{-1}^{+1} d\mu' \tag{69}$$

Then, in the self-adjoint transport equation

$$\mathcal{L}^{+}\mathcal{L}\phi - \mathcal{L}^{+}s = 0 \tag{70}$$

the operator

$$L = \mathcal{L}^{+}\mathcal{L} \tag{71}$$

becomes

$$\begin{aligned} L = & -\mu\frac{\partial}{\partial x}\left[\mu\frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2}\int_{-1}^{+1} d\mu'\right] \\ & + \Sigma_t\left[\mu\frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2}\int_{-1}^{+1} d\mu'\right] \\ & - \frac{\Sigma_s}{2}\int_{-1}^{+1}\left[\mu'\frac{\partial}{\partial x} + \Sigma_t - \frac{\Sigma_s}{2}\int_{-1}^{+1} d\mu''\right] d\mu' \end{aligned} \tag{72}$$

Collecting terms yields

$$\begin{aligned} L = & -\mu^2\frac{\partial^2}{\partial x^2} + \Sigma_t^2 - \Sigma_t\Sigma_s\int_{-1}^{+1} d\mu' \\ & + \frac{\Sigma_s}{2}\int_{-1}^{+1}(\mu-\mu')\frac{\partial}{\partial x}\,d\mu' + \frac{\Sigma_s^2}{4}\int_{-1}^{+1}\left[\int_{-1}^{+1} d\mu''\right] d\mu' \end{aligned} \tag{73}$$

Thus, the one-dimensional, self-adjoint transport equation for isotropic scatter becomes

$$\begin{aligned} & -\mu^2\frac{\partial^2\phi}{\partial x^2} + \Sigma_t^2\phi - \Sigma_s\Sigma_t\int_{-1}^{+1}\phi(x,\mu')\,d\mu' + \frac{\Sigma_s}{2}\int_{-1}^{+1}(\mu-\mu')\frac{\partial\phi'}{\partial x}\,d\mu' \\ & + \frac{\Sigma_s}{4}\int_{-1}^{+1}\left[\int_{-1}^{+1}\phi(x,\mu'')\,d\mu''\right]d\mu' = \mu\frac{\partial s}{\partial x} + \Sigma_t s - \frac{\Sigma_s}{2}\int_{-1}^{+1} s(x,\mu')\,d\mu' \end{aligned} \tag{74}$$

Since the integrals in the double integral term are independent

$$\int_{-1}^{+1} d\mu' = 2 \tag{75}$$

Thus

$$-\mu^2\frac{\partial^2\phi}{\partial x^2}+\Sigma_t^2\phi+\left(\frac{\Sigma_s^2}{2}-\Sigma_s\Sigma_t\right)\int_{-1}^{+1}\phi(x,\mu')\,d\mu'$$

$$+\frac{\Sigma_s}{2}\int_{-1}^{+1}(\mu-\mu')\frac{\partial\phi'}{\partial x}\,d\mu'=\mu\frac{\partial S}{\partial x}+\Sigma_t S-\frac{\Sigma_s}{2}\int_{-1}^{+1}S(x,\mu)\,d\mu' \qquad (76)$$

and

$$-\mu^2\frac{\partial^2\phi}{\partial x^2}+\Sigma_t^2\phi-\alpha\int_{-1}^{+1}\phi(x,\mu')\,d\mu'+\frac{\Sigma_s}{2}\int_{-1}^{+1}(\mu-\mu')\frac{\partial\phi'}{\partial x}\,d\mu'$$

$$=\mu\frac{\partial S}{\partial x}+\Sigma_t S-\frac{\Sigma_s}{2}\int_{-1}^{+1}S(x,\mu')\,d\mu' \qquad (77)$$

III. The Finite Element Equations

Finite Element Fundamentals

The finite element method is a numerical technique which divides the entire domain of the problem into simple polygonal structures called elements. The solution over this domain is approximated by a set of discrete values, called the nodal values, which are located at the nodes, or vertices, of the elements. The arrangement of nodes and elements is called a mesh. A typical mesh for a two-dimensional problem might look something like Fig. 1.

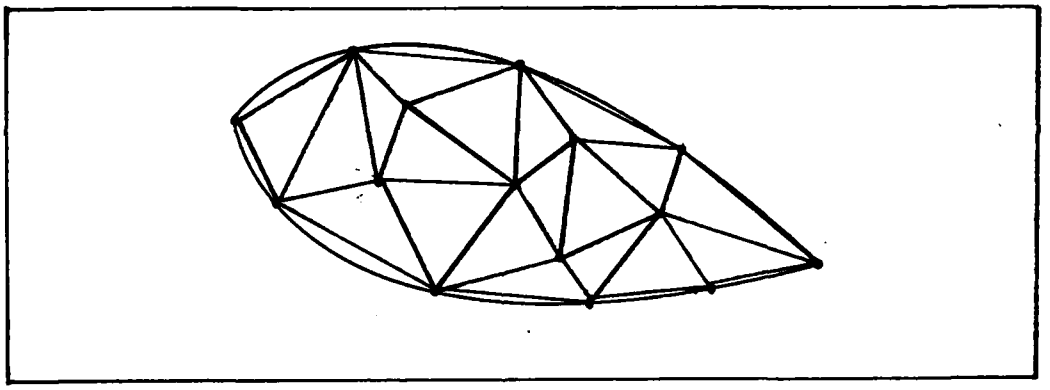


Fig. 1. Typical Finite Element Mesh

The set of nodal values is chosen so as to extremize the variational statement of the problem. They are computed by solving the set of simultaneous equations derived from the variational integral.

The solution is approximated within each element by

interpolating functions. If ϕ is the dependent variable, then over the x-y plane ϕ is approximated by

$$\phi(x,y) = \sum_i^n N_i(x_i y_i | x y)\,\phi_i \tag{78}$$

where n is the number of nodes. This is the finite element approximation. For this element, ϕ_i are the nodal values at the vertices of the element and N_i are the interpolating functions. The interpolating functions are usually linear, but higher order polynomials are sometimes used.

In the finite element approximation, the variational formulation has the form

$$I = \tfrac{1}{2}\underline{\phi}\,\underline{M}\,\underline{\phi} - \underline{\phi}\cdot\underline{b} \tag{79}$$

where b comes from the source term. Because the operator L was self-adjoint, $\underline{M}$ is symmetric and positive definite.

In the continuum formulation, the variation is taken with respect to the dependent variable. In the finite element method, the variation is taken with respect to the nodal values. In summation notation, the integral to be extremized is

$$I = \tfrac{1}{2}\sum_{jk}^{n}\phi_j M_{jk}\phi_k - \sum_i^n \phi_i b_i \tag{80}$$

Taking the variation with respect to the i'th node

$$\delta I = \sum_i^n \frac{\partial I}{\partial \phi_i}\delta\phi_i = \sum_i\left[\tfrac{1}{2}\left[\sum_k^n M_{ik}\phi_k + \sum_j^n \phi_j M_{ji}\right] - b_i\right]\delta\phi_i = 0 \tag{81}$$

Since M_{jk} is symmetric and since the indices are dummy, this

becomes

$$\delta I = \sum_i^{\tilde{N}} \left[\tfrac{1}{2} \left[\sum_j^{\tilde{N}} M_{ij} \phi_j + \sum_j^{\tilde{N}} M_{ij} \phi_j \right] - b_i \right] \delta \phi_i = 0 \qquad (82)$$

thus,

$$\frac{\partial I}{\partial \phi_i} = \sum_j^{\tilde{N}} M_{ij} \phi_j - b_i = 0 \qquad (83)$$

or

$$\frac{\partial I}{\partial \phi_i} = \underline{\underline{M}} \underline{\phi} - \underline{b} = 0 \qquad (84)$$

These are the simultaneous equations. The solution to them yields the nodal values.

As in any differential equation, the solution in the variational formulation is not unique until boundary conditions are specified. In the variational formulation, the boundary conditions are specified by the requirement that the variation be zero for any points under boundary conditions. When the problem is posed in finite elements, some of the nodes are necessarily on the boundary and no variation is allowed at these points.

Numbering the nodes such that all the interior nodes come first allows the variational formulation, in the absence of sources, to be restated as

$$I = \tfrac{1}{2} \left[\tilde{\underline{\phi}} \mid \tilde{\underline{\Psi}} \right] \left[\begin{array}{c|c} \underline{\underline{M}}_{11} & \underline{\underline{M}}_{12} \\ \hline \underline{\underline{M}}_{21} & \underline{\underline{M}}_{22} \end{array} \right] \left[\begin{array}{c} \underline{\phi} \\ \hline \underline{\Psi} \end{array} \right] \qquad (85)$$

where $\underline{\phi}$ is the vector of interior nodes and $\underline{\Psi}$ is the vector

of boundary nodes. Thus, the variation becomes

$$\delta I = \tfrac{1}{2}\left\{\tilde{\phi}\,\underline{\underline{M}}_{11} + \underline{\underline{M}}_{11}\phi + \underline{\underline{M}}_{12}\psi + \tilde{\psi}\,\underline{\underline{M}}_{21}\right\} = 0 \tag{86}$$

or, since the matrix is symmetric,

$$\delta I = \underline{\underline{M}}_{11}\phi + \underline{\underline{M}}_{12}\psi = 0 \tag{87}$$

and the simultaneous equations become

$$\underline{\underline{M}}_{11}\phi = -\underline{\underline{M}}_{12}\psi \tag{88}$$

The variational principle is good over any sub-domain of the problem domain. Specifically, it is good over any element. Thus,

$$I = \tfrac{1}{2}\sum_{ij}^{\varepsilon} \phi_i M_{ij} \phi_j \tag{89}$$

is still valid, where ε is the number of nodes on the element and M_{ij} is called the local matrix.

The global matrix is created by assembling the local matrices from every finite element. The process of assembly has three steps:

(1) Zero the global matrix.

(2) For each element in the local matrix, translate its local indices into the global indices, and add it to the proper global element.

(3) Do step two for every finite element.

Since each node is associated with only a few elements, the resulting global matrix is sparse.

One of the strengths of the finite element method is the ability to place elements, and therefore nodes, where desired. In many numerical techniques, such as finite difference, the nodes must be in a rectangular grid, and usually with uniform spacing. In these other techniques, a lot of computation must be done in regions of low interest just to get sufficient resolution in the area of high interest.

In the finite element method, new elements can be added to the mesh arbitrarily. The decision on where to place them is usually dependent upon how the solution looked with the original mesh. For instance, areas where the solution is rapidly varying are likely candidates for a finer mesh. Since the finite element method came from a variational formulation, another guide to placing new elements is the value of the variational integral over each element. This value is also known as the penalty, and the associated variational integral is the penalty function. Elements with the highest value are sub-divided into new elements and the problem is re-solved. This process can be automated with a computer routine, refining the mesh until some predetermined convergence criteria is reached.

The most commonly used shape for an element in the plane is a triangle. It is the element used in this study, and a brief review of its properties follows.

The finite element approximation is

$$\phi(x,y) = \sum_{i}^{3} N_i \phi_i \qquad (90)$$

The linear interpolating function is, over the x-u plane,

$$N_i = \frac{a_i x + b_i u + c_i}{2A} \tag{91}$$

Where, with i, j, and k as cyclic indices,

$$a_i = u_j - u_k \tag{92}$$

$$b_i = x_k - x_j \tag{93}$$

$$c_i = x_j u_k - x_k u_j \tag{94}$$

and

$$2A = \begin{vmatrix} 1 & 1 & 1 \\ x_i & x_j & x_k \\ u_i & u_j & u_k \end{vmatrix} \tag{95}$$

There is a natural coordinate system for triangles, called the triangular coordinate system (3:89). It is depicted in Fig. 2. The plane coordinates are related to the triangular coordinates by

$$\begin{bmatrix} 1 \\ x \\ u \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ u_1 & u_2 & u_3 \\ \phi_1 & \phi_2 & \phi_3 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{96}$$

Note that the relationship

$$\phi = \phi_1 L_1 + \phi_2 L_2 + \phi_3 L_3 \tag{97}$$

implies that, for linear interpolating functions,

$$L_i = N_i \tag{98}$$

The formula for integrating over the entire triangle is

$$\int_A L_1^p L_2^q L_3^r \, dA = \frac{p!\, q!\, r!}{(p+q+r+2)!} 2A \tag{99}$$

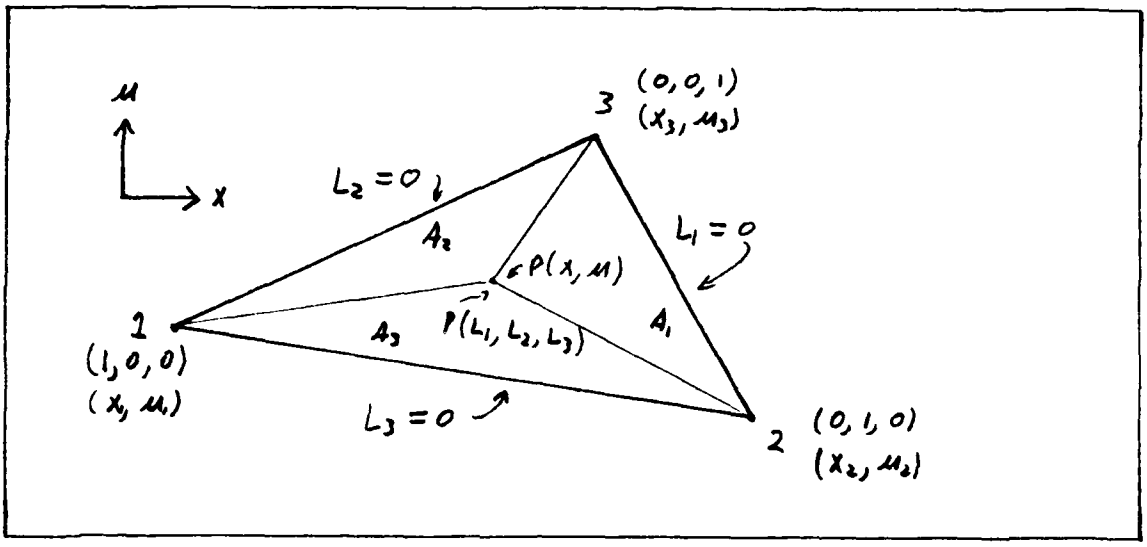


Fig. 2. Natural Triangular Coordinates

Two useful texts are The Finite Element Method for Engineers, by Kenneth H. Huebner (7), and An Analysis of the Finite Element Method, by George Fix and Gilbert Strang (6). The first text concentrates on applications, the second one on theory.

The Variational Integral

In this study, the finite element equations for triangular elements were derived for the homogeneous self-adjoint transport equation without sources. The integral to be extremized for this equation is

$$I = \tfrac{1}{2} \int \phi L \phi' \, d\mathcal{A} \tag{100}$$

or

$$I = \tfrac{1}{2} \int \left[-\mu^2 \phi \frac{\partial^2 \phi}{\partial x^2} + \Sigma_t^2 \phi^2 - \alpha \phi \int_{-1}^{+1} \phi(x, \mu') \, d\mu' + \frac{\Sigma_s}{2} \phi \int_{-1}^{+1} (\mu - \mu') \frac{\partial \phi(x, \mu')}{\partial x} \, d\mu' \right] d\mathcal{A} \tag{101}$$

Except for the limits of integration, this is also the integral to be extremized over each element. Specifically,

$$I = \tfrac{1}{2}\int\left[-\mu^2\phi\frac{\partial^2\phi}{\partial x^2} + \Sigma_t^2\phi^2 - \alpha\phi\int_{-1}^{+1}\phi(x,\mu')\,d\mu' + \frac{\Sigma_s}{2}\phi\int_{-1}^{+1}(\mu-\mu')\frac{\partial\phi(x,\mu')}{\partial x}\,d\mu'\right]dA \qquad (102)$$

where dA specifies integration over the area of the element and is

$$dA = dx\,d\mu \qquad (103)$$

The next step is to substitute the interpolating functions for ϕ. It is desirable to use linear interpolating functions for their computational efficiency. However, the class of admissible interpolating functions is limited by the highest occurring derivative in the extremization integral (7:79). Because of the second derivative in the first term, linear interpolating functions will not converge to the correct answer. An integration by parts nicely resolves this difficulty.

Taking the term with the second order derivative

$$I_1 = \tfrac{1}{2}\int\left[-\mu^2\phi\frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right)\right]dA \qquad (104)$$

and making the following definitions:

$$w = \phi \qquad dv = \frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right) \qquad (105)$$

thus,

$$dw = \frac{\partial\phi}{\partial x}dx \qquad v = \frac{\partial\phi}{\partial x} \qquad (106)$$

Therefore,

$$I_1 = -\tfrac{1}{2}\int \mu^2 \left[\phi \frac{\partial \phi}{\partial x}\Big|_{x_1}^{x_2} - \int \left(\frac{\partial \phi}{\partial x}\right)^2 dx \right] d\mu \qquad (107)$$

In a variational approach, the variation is restricted to being zero on the boundary. Thus, the first term in the integration by parts, called the surface term, is clearly zero whenever x_1 and x_2 are on the boundary. Since this integral is over a single element, and since many elements are interior to the region, x_1 and x_2 will not, in general, lie on the boundary. However, the value of the surface term has equal magnitude and opposite sign for adjacent elements. Consequently, when the global matrix is assembled, all the contributions from this term cancel. Thus,

$$I_1 = \tfrac{1}{2}\int \mu^2 \left(\frac{\partial \phi}{\partial x}\right)^2 dA \qquad (108)$$

It is convenient to split the extremization integral into five integrals, one for each term. They are

$$I_1 = \tfrac{1}{2}\int \mu^2 \left(\frac{\partial \phi}{\partial x}\right)^2 dA \qquad (109)$$

which will be referred to as the streaming term,

$$I_2 = \tfrac{1}{2}\int \Sigma_t^2 \phi^2 dA \qquad (110)$$

which will be referred to as the absorbing term,

$$I_3 = -\tfrac{1}{2}\alpha \int \phi \int_{-1}^{+1} \phi' d\mu' dA \qquad (111)$$

which will be referred to as the first scatter term,

$$I_4 = \tfrac{1}{2} \frac{\Sigma_s}{2} \int \varphi \int_{-1}^{+1} \mu \frac{\partial \psi'}{\partial x} d\mu' dA \tag{112}$$

which will be referred to as the second scatter term,

$$I_5 = -\tfrac{1}{2} \frac{\Sigma_s}{2} \int \varphi \int_{-1}^{+1} \mu' \frac{\partial \psi'}{\partial x} d\mu' dA \tag{113}$$

which will be referred to as the third scatter term.

There is an important distinction to be drawn among the five integrals. The first two, the streaming and absorption terms, are local, i.e., the integration is over only the immediate element. In the scatter terms, however, the additional integration over μ' implies involving other elements. These terms are non-local.

Most problems in which the finite element method is used have only local terms. The resulting global matrix is sparse and only one local matrix per element is needed. The resulting computational efficiency usually more than compensates for the relative complexity of the finite element approach.

In this problem, the situation is not so favorable, but since the non-localness is inherent in the tranport equation, other numerical techniques also face similar difficulties. In general, this property of the transport equation is a difficulty in any numerical technique. How it is handled in this approach will be described in the section on non-local terms.

The Local Terms

Starting with the streaming term

$$I_1 = \tfrac{1}{2}\int \mu^2 \left(\frac{\partial \phi}{\partial x}\right)^2 dA \tag{114}$$

and substituting for ϕ the finite element approximation with linear interpolating functions

$$\phi(x,\mu) = \sum_i^3 \phi_i N_i(x,\mu) \tag{115}$$

The term becomes

$$I_1 = \tfrac{1}{2}\int \mu^2 \left(\sum_i^3 \phi_i \frac{\partial N_i}{\partial x}\right)^2 dA \tag{116}$$

or

$$I_1 = \tfrac{1}{2}\int \mu^2 \left(\sum_i^3 \phi_i \frac{\partial N_i}{\partial x}\right)\left(\sum_j^3 \phi_j \frac{\partial N_j}{\partial x}\right) dA \tag{117}$$

and

$$I_1 = \tfrac{1}{2}\sum_{ij}^3 \phi_i \int \mu^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} dA\, \phi_j \tag{118}$$

Clearly then,

$$M_{ij}^{(\omega)} = \int \mu^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} dA \tag{119}$$

where $M_{ij}^{(\omega)}$ is the contribution to the local matrix from the streaming term. The interpolating functions are

$$N_i = \frac{1}{2A}(a_i x + b_i \mu + c_i) \tag{120}$$

Substituting and taking the derivative yields

$$M_{ij}^{(\omega)} = \frac{1}{4A^2}\int \mu^2 dA\, a_i a_j \tag{121}$$

This integral can be computed from the triangular integration formula. Substituting for

$$u = u_1 L_1 + u_2 L_2 + u_3 L_3 \tag{122}$$

and integrating yields

$$\int u^2 dA = \frac{A}{6}\left[u_1(u_1+u_2)+u_2(u_2+u_3)+u_3(u_3+u_1)\right] \tag{123}$$

Let

$$P = \left[u_1(u_1+u_2)+u_2(u_2+u_3)+u_3(u_3+u_1)\right] \tag{124}$$

thus,

$$M_{ij}^{(i)} = \frac{P}{24A}\, a_i a_j \tag{125}$$

where

$$a_i = u_j - u_k \tag{126}$$

and the i, j, and k are cyclic. Another form, which is optimized for computational efficiency, is

$$M_{ij}^{(\omega)} = \left(\sqrt{\frac{P}{24A}}\, a_i\right)\left(\sqrt{\frac{P}{24A}}\, a_j\right) \tag{127}$$

This is the form used in the computer program. A copy appears in Appendix B.

Similarly, where the absorption term

$$I_2 = \tfrac{1}{2}\int \Sigma_t^2 \phi^2 dA \tag{128}$$

becomes

$$I_2 = \tfrac{1}{2} \sum_{ij}^{3} \phi_i \, \Sigma_t^2 \int N_i N_j \, dA \, \phi_j \tag{129}$$

but a shortcut can be taken by recognizing the equivalence between the linear interpolating functions and the natural triangular coordinates

$$N_i = \frac{1}{2A} \left(a_i x + b_i x + c_i \right) = L_i \tag{130}$$

Substituting the natural triangular coordinates for the linear interpolating functions

$$M_{ij}^{(a)} = \Sigma_t^2 \int L_i L_j \, dA \tag{131}$$

From the triangular integration formula

$$\int L_i L_i \, dA = \frac{A}{6} \qquad \int L_i L_j \, dA = \frac{A}{12} \tag{132}$$

Thus, the local matrix from the absorption term is

$$M_{ij}^{(a)} = \frac{A}{12} \Sigma_t^2 \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{133}$$

The Non-Local Terms

Before considering the scatter terms in detail, the issue of their non-locality must be dealt with. In Fig. 3, a potential mesh over the x-u plane is displayed. The shaded triangle is the current local element and the region between the dotted lines is the area of integration specified by $dx\,du'$.

The integration over u' involves many elements other than the local one, a decided disadvantage. Now, not only do distant elements contribute to the value associated with the local element, but often only a portion of the element contributes. To account for this partial contribution would seem to require the existence of virtual elements (the parts of the real elements). Their existences would be temporary and their structure would be different for every local element. In addition, there are now many nodes involved (instead of just three), so the sparseness of the global matrix is greatly reduced.

In order to avoid these difficulties, a restriction on the structure of the mesh is imposed. If the numbered nodes from Fig. 3 were moved to line up on the dotted lines as in Fig. 4, then the integration over u' involves only complete non-local elements. This greatly simplifies the derivation of the finite element equations. In addition, the number of nodes involved is somewhat less, so the loss of sparseness in the global matrix is less severe. In this particular example, nine additional nodes are involved in the integration over u' before the numbered nodes are moved, but only seven afterwards.

The proposed restriction on the mesh structure is to align the elements into columns as in Fig. 5. This will be referred to as the columnar scheme. While there is some loss in flexibility of placement of elements, the figure shows

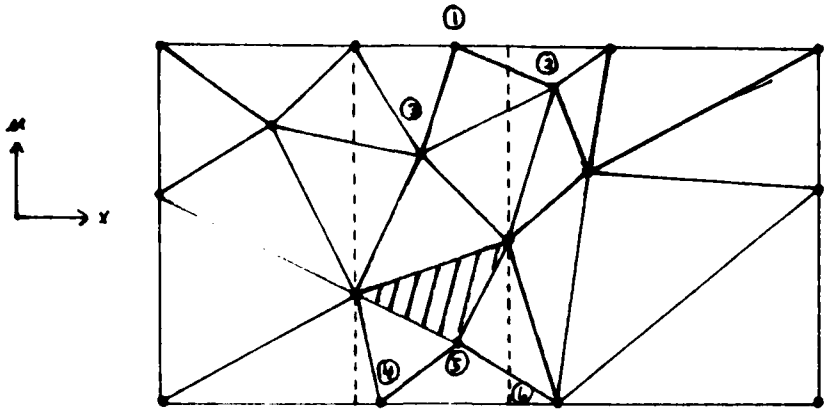


Fig. 3. Un-Restricted Finite Element Mesh

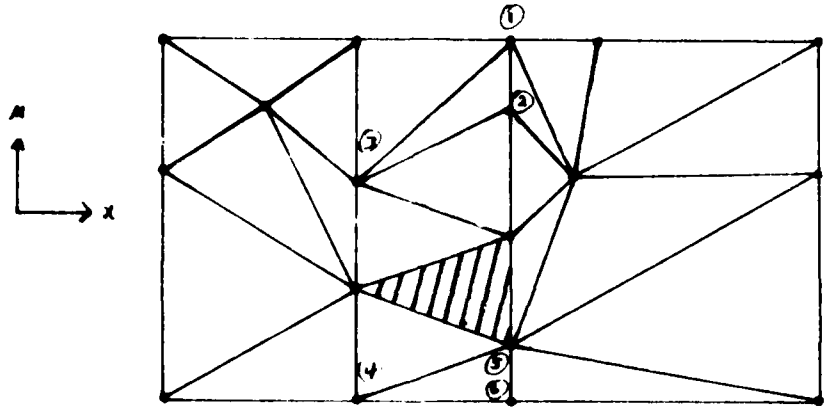


Fig. 4. Modified Finite Element Mesh

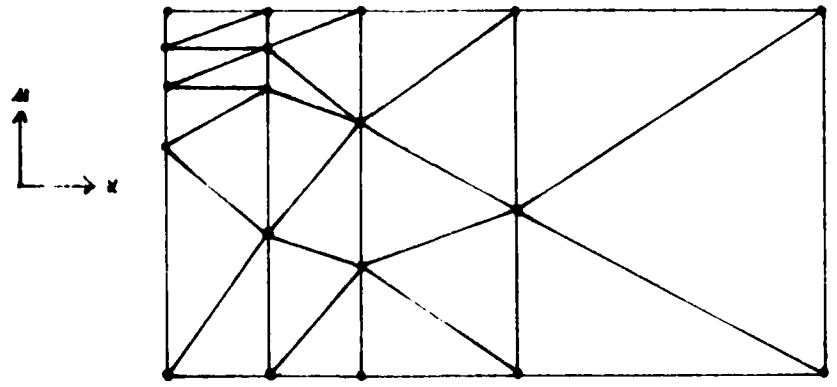


Fig. 5. Columnar Scheme Finite Element Mesh

that mesh structures are possible which still concentrate the elements in chosen regions.

The integration over the entire range of μ' is most conveniently done one non-local element at a time. The three scatter terms become

$$I_3 = -\tfrac{1}{2}\alpha \int \phi \int_{\underline{\mu}'}^{\bar{\mu}'} \phi' \, d\mu' \, dA \tag{134}$$

$$I_4 = \tfrac{1}{2}\frac{\Sigma_s}{2} \int \phi \int_{\underline{\mu}'}^{\bar{\mu}'} \mu \frac{\partial \phi'}{\partial x} \, d\mu' \, dA \tag{135}$$

$$I_5 = -\tfrac{1}{2}\frac{\Sigma_s}{2} \int \phi \int_{\underline{\mu}'}^{\bar{\mu}'} \mu' \frac{\partial \phi'}{\partial x} \, d\mu' \, dA \tag{136}$$

where $\bar{\mu}'$ is the upper edge of the non-local element and $\underline{\mu}'$ is the lower edge. Note that both $\bar{\mu}'$ and $\underline{\mu}'$ are functions of x. These integrals are then evaluated for every non-local element in the column containing the local element.

Since both the local and non-local elements may point in either direction, there are clearly four cases. The cases and the relevant coordinate system are summarized in Fig. 6.

The finite element approximation for the three scatter terms is a combination of the forms used in the streaming and absorption terms. For ϕ,

$$\phi(x,\mu) = \sum_i^3 L_i \phi_i \tag{137}$$

while for ϕ',

$$\phi'(x,\mu) = \sum_j^3 N_j' \phi_j' \tag{138}$$

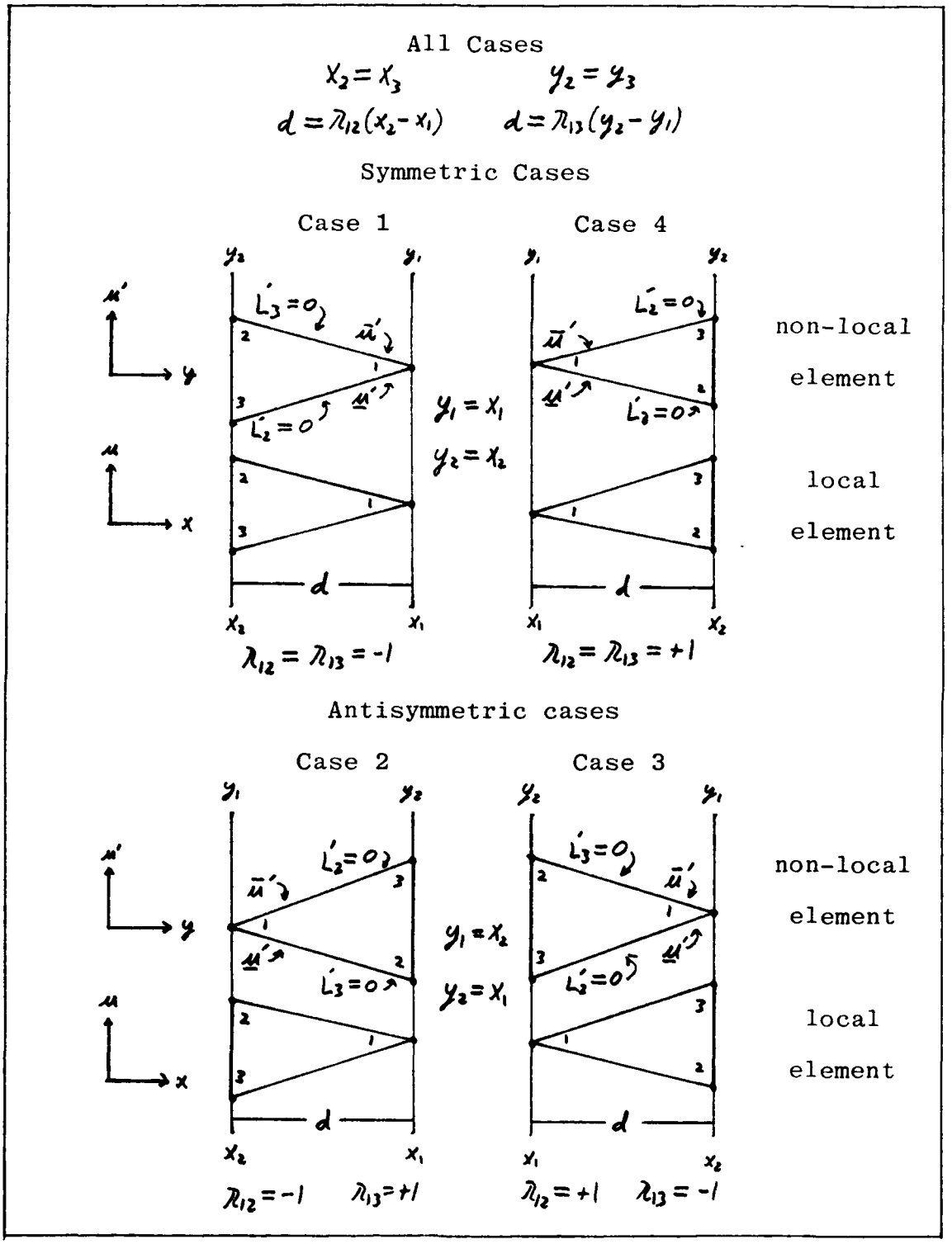


Fig. 6. Local and Non-local Element Cases

where

$$N_j' = \frac{1}{2A'}\left(a_j' x + b_j' u' + c_j'\right) \tag{139}$$

with

$$a_j' = u_k' - u_j' \tag{140}$$

$$b_j' = x_i - x_k \tag{141}$$

$$c_j' = x_k u_i' - x_i u_k' \tag{142}$$

and where A' is the area of the non-local element. Thus,

$$\frac{\partial \phi'}{\partial x} = \frac{1}{2A'} a_j' \tag{143}$$

For the three scatter terms

$$I_3 = -\tfrac{1}{2}\alpha \int \phi(x,u) \int_{u'}^{\bar{u}'} \phi'(x,u')\, du'\, dA \tag{144}$$

$$I_4 = \tfrac{1}{2}\frac{\varepsilon_s}{2} \int \phi(x,u) \int_{u'}^{\bar{u}'} u \frac{\partial \phi'(x,u')}{\partial x}\, du'\, dA \tag{145}$$

$$I_5 = -\tfrac{1}{2}\frac{\varepsilon_s}{2} \int \phi(x,u) \int_{u'}^{\bar{u}'} u' \frac{\partial \phi'(x,u')}{\partial x}\, du'\, dA \tag{146}$$

The following finite element approximations are made:

$$\phi(x,u) = \sum_i^3 L_i \phi_i \tag{147}$$

$$\phi'(x,u') = \sum_j^3 N_j' \phi_j' \tag{148}$$

where

$$N_j' = \frac{1}{2A'}\left(a_j' x + b_j' u' + c_j'\right) \tag{149}$$

Thus,

$$\frac{\partial \phi'}{\partial x} = \frac{1}{2A'} a_j' \tag{150}$$

Therefore, the matrices for each of the three scatter terms become

$$M_{ij}^{(3)} = -\alpha \frac{1}{2A'} \int L_i \int_{\underline{u}'}^{\bar{u}'} \left(a_j' x + b_j' u' + c_j'\right) du' dA \tag{151}$$

$$M_{ij}^{(4)} = \frac{\Sigma_s}{2} \frac{1}{2A'} \int L_i \int_{\underline{u}'}^{\bar{u}'} u\, a_j'\, du' dA \tag{152}$$

$$M_{ij}^{(5)} = -\frac{\Sigma_s}{2} \frac{1}{2A'} \int L_i \int_{\underline{u}'}^{\bar{u}'} u'\, a_j'\, du' dA \tag{153}$$

Integrating over u'

$$M_{ij}^{(3)} = -\alpha \frac{1}{2A'} \int L_i \left[\left(a_j' x + c_j'\right)\left(\bar{u}' - \underline{u}'\right) + \frac{1}{2} b_j' \left(\bar{u}'^2 - \underline{u}'^2\right)\right] dA \tag{154}$$

$$M_{ij}^{(4)} = \frac{\Sigma_s}{2} \frac{1}{2A'} \int L_i\, u\, a_j' \left(\bar{u}' - \underline{u}'\right) dA \tag{155}$$

$$M_{ij}^{(5)} = -\frac{\Sigma_s}{2} \frac{1}{2A'} \int L_i \frac{1}{2} a_j' \left(\bar{u}'^2 - \underline{u}'^2\right) dA \tag{156}$$

or

$$M_{ij}^{(3)} = -\alpha \frac{1}{2A'} \int L_i \left(\bar{u}' - \underline{u}'\right)\left[a_j' x + c_j' + \frac{1}{2} b_j \left(\bar{u}' + \underline{u}'\right)\right] dA \tag{157}$$

$$M_{ij}^{(4)} = \frac{\Sigma_s}{2} \frac{1}{2A'} \int L_i \left(\bar{u}' - \underline{u}'\right) a_j' u\, dA \tag{158}$$

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{1}{2A'}\int L_i(\bar{u}'-\underline{u}')\frac{1}{2}a_j'(\bar{u}'+\underline{u}')\,dA \qquad (159)$$

The upper and lower limits of integration, $\bar{u}'$ and $\underline{u}'$, must be evaluated, depending upon the case. For Cases 1 and 3, the lower limit ($\underline{u}'$) is specified by

$$L_2' = 0 \qquad (160)$$

thus,

$$\begin{bmatrix} 1 & 1 & 1 \\ y_1 & y_3 & y \\ u_1' & u_3' & \underline{u}' \end{bmatrix}\begin{bmatrix} L_1' \\ L_3 \\ -1 \end{bmatrix} = 0 \qquad (161)$$

for this to be equal to zero, the determinant must be equal to zero. Thus,

$$\begin{vmatrix} 1 & 1 & 1 \\ y_1 & y_3 & y \\ u_1' & u_3' & \underline{u}' \end{vmatrix} = 0 \qquad (162)$$

which implies

$$\underline{u}'(y_3-y_1) - u_3'(y-y_1) + u_1'(y-y_3) = 0 \qquad (163)$$

and

$$\underline{u}' = \frac{u_3'-u_1'}{y_3-y_1}y + \frac{u_1'y_3-u_3'y_1}{y_3-y_1} \qquad (164)$$

However, x and y have the same range and sense, and since $y_2 = y_3$

$$\underline{u}' = \frac{u_3'-u_1'}{y_2-y_1}x + \frac{u_1'y_2-u_3'y_1}{y_2-y_1} \qquad (165)$$

By symmetry, the upper limit is

$$\bar{u}' = \frac{u_2' - u_1'}{y_2 - y_1} x + \frac{u_1' y_2 - u_2' y_1}{y_2 - y_1} \tag{166}$$

For Cases 2 and 4, the lower limit is specified by

$$L_3' = 0 \tag{167}$$

thus,

$$\begin{vmatrix} 1 & 1 & 1 \\ y_1 & y_2 & y \\ u_1' & u_2' & \underline{u}' \end{vmatrix} = 0 \tag{168}$$

which implies

$$\underline{u}'(y_2 - y_1) - u_2'(y - y_1) + u_1'(y - y_2) = 0 \tag{169}$$

and

$$\underline{u}' = \frac{u_2' - u_1'}{y_2 - y_1} y + \frac{u_1' y_2 - u_2' y_1}{y_2 - y_1} \tag{170}$$

However, x and y have the same range and sense, so this is just

$$\underline{u}' = \frac{u_2' - u_1'}{y_2 - y_1} x + \frac{u_1' y_2 - u_2' y_1}{y_2 - y_1} \tag{171}$$

So again, by symmetry, the upper limit is

$$\bar{u}' = \frac{u_3' - u_1'}{y_2 - y_1} x + \frac{u_1' y_2 - u_3' y_1}{y_2 - y_1} \tag{172}$$

In the local matrix for the scatter terms, $\bar{u}'$ and $\underline{u}'$ appear in only two combinations

$$\bar{u}' - \underline{u}' \tag{173}$$

and

$$\bar{u}' + \underline{u}' \tag{174}$$

The first combination becomes, for Cases 1 and 3,

$$\bar{u}' - \underline{u}' = \frac{1}{y_2 - y_1}\left[(u_2' - u_1')x + (u_1' y_2 - u_2' y_1) - (u_3' - u_1')x - (u_1' y_2 - u_3' y_1)\right] \tag{175}$$

or

$$\bar{u}' - \underline{u}' = \frac{1}{y_2 - y_1}\left[(u_3' - u_2')(x - y_1)\right] \tag{176}$$

which can be simplified through

$$\bar{u}' - \underline{u}' = \frac{1}{(y_2 - y_1)^2}\left[(u_3' - u_2')(y_2 - y_1)\right](x - y_1) \tag{177}$$

to

$$\bar{u}' - \underline{u}' = -\frac{2A'}{d^2}(x - y_1) \tag{178}$$

For Cases 2 and 4,

$$\bar{u}' - \underline{u}' = \frac{1}{y_2 - y_1}\left[(u_3' - u_1')x - (u_2' - u_1')x + (u_1' y_2 - u_3' y_1) - (u_1' y_2 - u_2' y_1)\right] \tag{179}$$

as before, this is

$$\bar{u}' - \underline{u}' = \frac{2A'}{d^2}(x - y_1) \tag{180}$$

Therefore,

$$\bar{u}' - \underline{u}' = \lambda_{13}\frac{2A'}{d^2}(x - y_1) \tag{181}$$

where λ_{13} is negative one for Cases 1 and 3, and positive one for Cases 2 and 4.

The second combination of the integration limits is

$$\bar{u}' + \underline{u}' \tag{182}$$

Again, considering Cases 1 and 3 first, this becomes

$$\bar{u}'+\underline{u}' = \frac{1}{y_2-y_1}\left[(u_3'-u_1')x + (u_3'-u_1')x + (u_1'y_2 - u_2'y_1) + (u_1'y_2 - u_3'y_1)\right] \tag{183}$$

or

$$\bar{u}'+\underline{u}' = \frac{1}{y_2-y_1}\left[(u_3'+u_2'-2u_1')x - (u_3'+u_2'-2u_1'\frac{y_2}{y_1})y_1\right] \tag{184}$$

Let

$$\ell = (u_3' + u_2' - 2u_1') \tag{185}$$

and

$$k = (u_3' + u_2' - 2u_1'\frac{y_2}{y_1}) \tag{186}$$

Thus,

$$\bar{u}'+\underline{u}' = -\frac{1}{d}(\ell x - k y_1) \tag{187}$$

For Cases 2 and 4,

$$\bar{u}'+\underline{u}' = \frac{1}{y_2-y_1}\left[(u_3'-u_1')x + (u_2'-u_1')x + (u_1'y_2 - u_3'y_1) + (u_1'y_2 - u_2'y_1)\right] \tag{188}$$

or

$$\bar{u}'+\underline{u}' = \frac{1}{y_2-y_1}\left[(u_3'+u_2'-2u_1')x + (u_3'+u_2'-2u_1'\frac{y_2}{y_1})y_1\right] \tag{189}$$

and making the same substitution

$$\bar{u}' + u' = \frac{1}{d}(\ell x - k y_1) \tag{190}$$

Thus,

$$\bar{u}' + u' = \frac{n_{13}}{d}(\ell x - k y_1) \tag{191}$$

Therefore, the matrices for each of the three scatter terms become

$$M_{ij}^{(3)} = -\alpha n_{13} \frac{1}{d^2} \int L_i(x-y_1)\left[a_j' x + c_j' + \frac{1}{2}\frac{n_{13}}{d} b_j'(\ell x - k y_1)\right] dA \tag{192}$$

$$M_{ij}^{(4)} = \frac{\Sigma_s}{2} n_{13} \frac{1}{d^2} \int L_i(x-y_1)\, a_j' u\, dA \tag{193}$$

$$M_{ij}^{(6)} = -\frac{\Sigma_s}{2} n_{13} \frac{1}{d^2} \int L_i(x-y_1) \frac{1}{2}\frac{n_{13}}{d} a_j'(\ell x - k y_1)\, dA \tag{194}$$

Note that there is a lot of parallelism among these three expressions. Later mathematical manipulations can be reduced by a judicious rearranging of the terms within each expression. Thus,

$$\begin{aligned} M_{ij}^{(3)} = &-\alpha n_{13} \frac{1}{d^2} c_j' \int L_i(x-y_1)\, dA \\ &-\alpha n_{13} \frac{1}{d^2} a_j' \int L_i(x-y_1)\, x\, dA \\ &-\alpha \frac{1}{2} \frac{1}{d^3} b_j' \int L_i\left[\ell x^2 - (\ell y_1 + k y_1) x + k y_1^2\right] dA \end{aligned} \tag{195}$$

and

$$M_{ij}^{(4)} = \frac{\Sigma_s}{2} n_{13} \frac{1}{d^2} a_j' \int L_i(x-y_1)\, u\, dA \tag{196}$$

and

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{1}{2}\frac{1}{d^3} a_j' \int L_i \left[\ell x^2 - (\ell + k) y_1 x + k y_1^2 \right] dA \qquad (197)$$

Before proceeding to evaluate each scatter term, it is convenient to build a table of integrals. The five integrals, which are based upon the triangular integration formula, are derived in Appendix A and are listed in Table I.

A note about the notation used in Table I is in order. The vector of interpolating coefficients is

$$\underline{L} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \left[L_i \right] \qquad i = 1, 2, 3 \qquad (198)$$

The area, A, is for the local element.

Because of the parallelism in the form of the scatter terms, it seems most convenient to evaluate the simplest first. Starting, then, with the second scatter term, its matrix can be rewritten as

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{1}{d^2} a_j' \left\{ \int \underline{L}\, x\, u\, dA - \int \underline{L}\, y_1\, u\, dA \right\} \qquad (199)$$

which becomes

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60 d^2} a_j' \left\{ x_1 \underline{m}_1 + x_2 \underline{m}_b - y_1 \underline{m}_a \right\} \underline{u} \qquad (200)$$

For the symmetric cases, Cases 1 and 4, where $y_1 = x_1$

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60 d^2} a_j' \left\{ x_1 (\underline{m}_1 - \underline{m}_a) + x_2 \underline{m}_b \right\} \underline{u} \qquad (201)$$

or

Table I

Five Element Integrals

$$\underline{\underline{m}}_a = \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix}$$

$$\underline{\underline{m}}_1 = \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \qquad \underline{\underline{m}}_b = \begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix}$$

$$\underline{\underline{m}}_a = \underline{\underline{m}}_1 + \underline{\underline{m}}_b$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\int \underline{L}\, dA = \frac{A}{60}\, \underline{\underline{m}}_a \underline{I}$$

$$\int \underline{L}\, x\, dA = \frac{A}{60}\, \underline{\underline{m}}_a \underline{x}$$

$$\int \underline{L}\, u\, dA = \frac{A}{60}\, \underline{\underline{m}}_a \underline{u}$$

$$\int \underline{L}\, x^2\, dA = \frac{A}{60} \left\{ x_1 \underline{\underline{m}}_1 + x_2 \underline{\underline{m}}_b \right\} \underline{x}$$

$$\int \underline{L}\, x u\, dA = \frac{A}{60} \left\{ x_1 \underline{\underline{m}}_1 + x_2 \underline{\underline{m}}_b \right\} \underline{u}$$

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60d^2} a_j' \{(x_2 - x_1) \underline{m}_b\} \underline{u} \tag{202}$$

Thus,

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60d^2} a_j' \pi_{12} d \underline{m}_b \underline{u} \tag{203}$$

which is

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \frac{A}{60d} a_j' \underline{m}_b \underline{u} \tag{204}$$

Explicitly,

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \frac{A}{60d} \begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} a_j' \tag{205}$$

Similarly, for the antisymmetric cases, where $y_1 = x_2$

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60d^2} a_j' \{(x_1 - x_2) \underline{m}_1\} \underline{u} \tag{206}$$

or

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \pi_{13} \frac{A}{60d^2} a_j' \pi_{34} d \underline{m}_1 \underline{u} \tag{207}$$

Thus,

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \frac{A}{60d} a_j' \underline{m}_1 \underline{u} \tag{208}$$

Explicitly,

$$M_{ij}^{(4)} = \frac{\varepsilon_s}{2} \frac{A}{60d} \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} a_j' \tag{209}$$

Evaluating the third scatter term next, its matrix can be rewritten as

$$M_{ij}^{(s)} = -\frac{\varepsilon_s}{2}\frac{1}{d^3}a_j'\int\frac{1}{2}\underline{L}\left(\ell x^2 - 2c\,y_1 x + k y_1^2\right)dA \qquad (210)$$

where

$$c = \frac{1}{2}(\ell + k) \qquad (211)$$

Expanding

$$c\,y_1 = (\mu_3' + \mu_2')y_1 - \mu_1'(y_1 + y_2) \qquad (212)$$

Substituting for ℓ and k, and evaluating the integrals yields

$$\begin{aligned} M_{ij}^{(s)} = -\frac{\varepsilon_s}{2}\frac{A}{10d^3}a_j\frac{1}{2}\Big\{ & -2\mu_1'\left[(x_1\underline{m}_1 + x_2\underline{m}_2)\underline{x} - (y_1+y_2)\underline{m}_a\underline{x} + y_1y_2\underline{m}_a\underline{I}\right] \\ & + \mu_2'\left[(x_1\underline{m}_1 + x_2\underline{m}_2)\underline{x} - 2y_1\underline{m}_a\underline{x} + y_1^2\underline{m}_a\underline{I}\right] \\ & + \mu_3'\left[(x_1\underline{m}_1 + x_2\underline{m}_2)\underline{x} - 2y_1\underline{m}_a\underline{x} + y_1^2\underline{m}_a\underline{I}\right]\Big\} \end{aligned} \qquad (213)$$

The coefficient of μ_1' is independent of case. Working with just that term, it becomes, for all cases

$$2\mu_1'\left[-(x_1\underline{m}_1 + x_2\underline{m}_2)\underline{x} + (x_1+x_2)\underline{m}_a\underline{x} - x_1x_2\underline{m}_a\underline{I}\right] \qquad (214)$$

Rearranging and combining terms

$$2\mu_1'\left[x_1\underline{m}_2\underline{x} + x_2\underline{m}_1\underline{x} - x_1x_2\underline{m}_a\underline{I}\right] \qquad (215)$$

Expanding the matrices, and then contracting them on the basis of $x_2 = x_3$ yields

$$2\mu_1' \left\{ x_1 \begin{bmatrix} 4 & 6 \\ 3 & 12 \\ 3 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + x_2 \begin{bmatrix} 6 & 4 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - x_1 x_2 \begin{bmatrix} 10 & 10 \\ 5 & 15 \\ 5 & 15 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \tag{216}$$

Rearranging around the quadratic terms in x yields

$$2\mu_1' \begin{Bmatrix} 4x_1^2 - 8x_1x_2 + 4x_2^2 \\ 3x_1^2 - 6x_1x_2 + 3x_2^2 \\ 3x_1^2 - 6x_1x_2 + 3x_2^2 \end{Bmatrix} \tag{217}$$

But this is just

$$2\mu_1' \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix} d^2 \tag{218}$$

The coefficients of μ_2' and μ_3' are identical. Therefore, only one needs to be evaluated. However, they are case dependent. Thus, for the symmetric cases

$$\mu_2' \left[(x_1 \underline{\underline{m}}_1 + x_2 \underline{\underline{m}}_b) \underline{x} - 2x_1 \underline{\underline{m}}_a \underline{x} + x_1^2 \underline{\underline{m}}_a \underline{\underline{I}} \right] \tag{219}$$

or

$$\mu_2' \left[(x_2 - x_1) \underline{\underline{m}}_b \underline{x} + x_1 \underline{\underline{m}}_a (x_1 \underline{\underline{I}} - \underline{x}) \right] \tag{220}$$

Expanding the matrices as before

$$\mu_2' \left\{ (x_2 - x_1) \begin{bmatrix} 4 & 6 \\ 3 & 12 \\ 3 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + x_1 \begin{bmatrix} 10 & 10 \\ 5 & 15 \\ 5 & 15 \end{bmatrix} \begin{bmatrix} x_1 - x_1 \\ x_1 - x_2 \end{bmatrix} \right\} \tag{221}$$

Rearranging around

$$u_3' \begin{Bmatrix} (x_2-x_1)(4x_1+6x_2-10x_1) \\ (x_2-x_1)(3x_1+12x_2-15x_1) \\ (x_2-x_1)(3x_1+12x_2-15x_1) \end{Bmatrix} = u_3' \begin{Bmatrix} (x_2-x_1)^2\, 6 \\ (x_2-x_1)^2\, 12 \\ (x_2-x_1)^2\, 12 \end{Bmatrix} \tag{222}$$

which is

$$2u_3' \begin{bmatrix} 3 \\ 6 \\ 6 \end{bmatrix} d^2 \tag{223}$$

Similarly, for the antisymmetric cases

$$u_3' \left[(x_1 \underline{m}_1 \underline{x} + x_2 \underline{m}_b \underline{x}) - 2x_2 \underline{m}_a \underline{x} + x_2^2 \underline{m}_a \underline{I} \right] \tag{224}$$

or

$$u_3' \left[(x_1 - x_2) \underline{m}_1 \underline{x} + x_2 \underline{m}_a (x_2 \underline{I} - \underline{x}) \right] \tag{225}$$

Again, expanding the matrices

$$u_3' \left\{ (x_1-x_2) \begin{bmatrix} 6 & 4 \\ 2 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + x_2 \begin{bmatrix} 10 & 10 \\ 5 & 15 \\ 5 & 15 \end{bmatrix} \begin{bmatrix} x_2-x_1 \\ x_2-x_2 \end{bmatrix} \right\} \tag{226}$$

Rearranging around $x_1 - x_2$ yields

$$u_2' \begin{Bmatrix} (x_1-x_2)(6x_1+4x_2-10x_2) \\ (x_1-x_2)(2x_1+3x_2-5x_2) \\ (x_1-x_2)(2x_1+3x_2-5x_2) \end{Bmatrix} \tag{227}$$

which is just

$$2u_2' \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} d^2 \tag{228}$$

Therefore, for the symmetric case

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{A}{60d}\, a_j' \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \underline{u}' \tag{229}$$

or

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{A}{60d}\, a_j' \underline{m}_c \underline{u}' \tag{230}$$

where

$$\underline{m}_c = \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \tag{231}$$

And likewise, for the antisymmetric case

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{A}{60d}\, a_j' \begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix} \underline{u}' \tag{232}$$

or

$$M_{ij}^{(5)} = -\frac{\varepsilon_s}{2}\frac{A}{60d}\, a_j' \underline{m}_d \underline{u}' \tag{233}$$

where

$$\underline{m}_d = \begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix} \tag{234}$$

The first scatter term is most easily evaluated one piece at a time. The first piece

$$-\alpha \lambda_{13} \frac{1}{d^2}\, c_j' \int \underline{L}(x - y_1)\, dA \tag{235}$$

becomes

$$-\alpha \pi_{13} \frac{A}{60d^2} C_j' \underline{\underline{m}}_a (X - y_1 \underline{\underline{I}}) \tag{236}$$

For the symmetric cases

$$-\alpha \pi_{13} \frac{A}{60d^2} C_j' \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix} \begin{bmatrix} x_1 - x_1 \\ x_2 - x_1 \\ x_3 - x_1 \end{bmatrix} \tag{237}$$

or

$$-\alpha \pi_{13} \frac{A}{60d^2} C_j' \begin{bmatrix} 0 & 5 & 5 \\ 0 & 10 & 5 \\ 0 & 5 & 10 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \pi_{12} d \tag{238}$$

or

$$-\alpha \frac{A}{60d} C_j' \underline{\underline{h}}_2 \underline{\underline{I}} \tag{239}$$

For the antisymmetric cases

$$-\alpha \pi_{13} \frac{A}{60d^2} C_j' \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix} \begin{bmatrix} x_1 - x_2 \\ x_2 - x_2 \\ x_3 - x_2 \end{bmatrix} \tag{240}$$

or

$$-\alpha \pi_{13} \frac{A}{60d^2} C_j' \begin{bmatrix} 10 & 0 & 0 \\ 5 & 0 & 0 \\ 5 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \pi_{34} d \tag{241}$$

or

$$-\alpha \frac{A}{60d} C_j' \underline{\underline{h}}_1 \underline{\underline{I}} \tag{242}$$

The second piece

$$-\alpha \bar{n}_{13} \frac{1}{d^2} a_j' \int_{\perp} (x - y_1) \times dA \tag{243}$$

by symmetry with the second scatter term is

$$-\alpha \frac{A}{60d} \begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} a_j' \tag{244}$$

or

$$-\alpha \frac{A}{60d} a_j' \underline{\underline{m}}_0 \underline{x} \tag{245}$$

for the symmetric cases, and

$$-\alpha \frac{A}{60d} \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} a_j' \tag{246}$$

or

$$-\alpha \frac{A}{60d} a_j' \underline{\underline{m}}_1 \underline{x} \tag{247}$$

for the antisymmetric cases.

The third piece

$$-\alpha \frac{1}{d^3} b_j' \int \frac{1}{2} \perp \left[\ell x^2 - 2c y_1 x + k y_1^2 \right] dA \tag{248}$$

by symmetry with the third scatter term is

$$-\alpha \frac{A}{60d} b_j' \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \underline{\mu}' \tag{249}$$

or

$$-\alpha \frac{A}{60d} b_j' \underline{\underline{m}}_c \underline{u}' \tag{250}$$

for the symmetric cases, and

$$-\alpha \frac{A}{60d} b_j' \begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix} \underline{u}' \tag{251}$$

or

$$-\alpha \frac{A}{60d} b_j' \underline{\underline{m}}_d \underline{u}' \tag{252}$$

for the antisymmetric cases.

Putting the three pieces together for the symmetric cases yields

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \left[\underline{\underline{m}}_b \underline{x} a_j' + \underline{\underline{m}}_c \underline{u}' b_j' + \underline{b} \cdot \underline{\underline{I}} c_j' \right] \tag{253}$$

Expanding the matrices

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \left\{ \begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} a_j' + \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} b_j' + \begin{bmatrix} 0 & 5 & 5 \\ 0 & 10 & 5 \\ 0 & 5 & 10 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} c_j' \right\} \tag{254}$$

Since $x_2 = x_3$, the lower right four elements in the first matrix can be rearranged. Similarly, all of the elements in the last matrix can be freely rearranged. Thus,

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \left\{ \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} a_j' + \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} b_j' + \begin{bmatrix} 4 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} c_j' \right\} \tag{255}$$

or

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \underline{\underline{m}}_c \left[a_j' x + b_j' \underline{u}' + c_j' \underline{I} \right] \tag{256}$$

The bracketed term can be evaluated by recognizing that the natural triangular coordinates in the non-local element are

$$L_j' = \frac{1}{2A'} \left[a_j' x + b_j' u' + c_j' \right] \tag{257}$$

which, for a point at its corresponding node, is equal to one

$$1 = \frac{1}{2A'} \left[a_j' x_j + b_j' u_j' + c_j' \right] \tag{258}$$

but at any other node is equal to zero

$$0 = \frac{1}{2A'} \left[a_j' x_i + b_j' u_i' + c_j' \right] \qquad i \neq j \tag{259}$$

Thus, the bracketed term is

$$\left[a_j' x_i + b_j' u_i' + c_j' \right] = 2A' \tag{260}$$

and therefore, for the symmetric case, the matrix for the first scatter term is

$$M_{ij}^{(3)} = -\alpha \frac{AA'}{30d} \underline{\underline{m}}_c \tag{261}$$

Putting the three pieces together for the antisymmetric cases yields

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \left[\underline{\underline{m}}_1 x a_j' + \underline{\underline{m}}_2 \underline{u}' b_j' + \underline{\underline{b}}_1 \underline{I} c_j' \right] \tag{262}$$

Expanding the matrices

$$M_{ij}^{(3)} = -\alpha \frac{A}{60d} \left\{ \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} a_j' + \begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_1' \\ u_2' \\ u_3' \end{bmatrix} b_j' + \begin{bmatrix} 10 & 0 & 0 \\ 5 & 0 & 0 \\ 5 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} c_j' \right\} \quad (263)$$

While it is still possible to rearrange the terms in the matrices corresponding to $x_2 = x_3$, it does no good because the x_1 term is no longer equal to the u_1' term in the second matrix as it was in the symmetric cases. Additionally, this author was unable to simplify this expression to any significant degree.

A simple answer was obtained by a different derivation and it is included in Appendix G. However, it does not build on the derivation presented here. Nonetheless, it does obtain a useful answer, one which was confirmed independently by Dr. Shankland, and which is in agreement with the answer one might anticipate from the results in the symmetric case. Assuming, for the moment, that this answer is correct, the matrix for the first scatter term for the antisymmetric case is

$$M_{ij}^{(3)} = -\alpha \frac{AA'}{30d} \underline{\underline{m}}_d \quad (264)$$

where

$$\underline{\underline{m}}_d = \begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix} \quad (265)$$

IV. The Numerical Solution

Benchmark Problem

A picture of the benchmark problem used in this study is shown in Fig. 7.

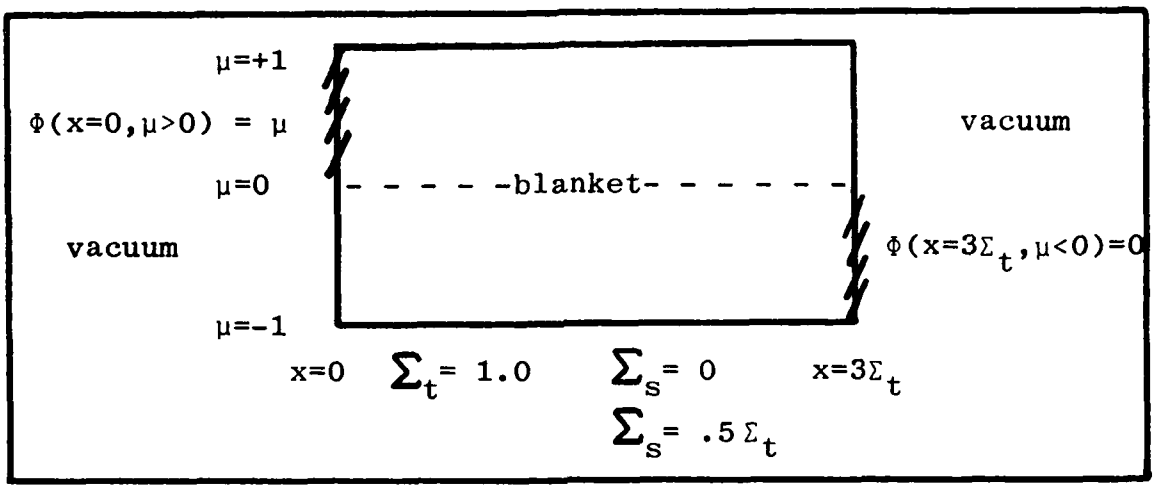


Fig. 7. Benchmark Problem

The benchmark problem consisted of a mono-energetic source of particles which impinged from the left upon a blanket three mean free paths thick. The source was Lambertian, and constituted the first of two boundary conditions. The blanket was surrounded on both sides by vacuum and an albedo of zero was assumed so the returning flux from the right was also zero. This was the second boundary condition. The total macroscopic cross-section was chosen as unity so that it could serve as a scaling factor. The scattering macroscopic cross-section was either zero, which eliminated the

scattering terms, or half the total cross-section. Both macroscopic cross-sections were constant throughout the blanket. There were no inbedded sources in the blanket.

Algorithm

The algorithm for the benchmark problem follows the usual finite element approach, except for the handling of the scatter terms. Specifically:

(1) Partition the global matrix into four sub-matrices according to the interior and boundary nodes.

(2) Zero the global matrix.

(3) Define the mesh.

(4) Compute the contribution to the local matrix from the streaming term and assemble it into the global matrix.

(5) Compute the contribution to the local matrix from the absorption term and assemble it into the global matrix.

(6) Save the contributions from the streaming and absorption terms.

(7) For each non-local element in the column:
    (a) Compute the contribution to the local matrix from the first scatter term and assemble it into the global matrix.
    (b) Compute the contribution to the local matrix from the second scatter term and assemble it into the global matrix.
    (c) Compute the contribution to the local matrix from the third scatter term and assemble it into the global matrix.
    (d) Save the contributions from the three scatter terms.

(8) Compute the force vector.

(9) Solve the system of simultaneous equations.

(10) Compute the element penalties and the global penalty.

(11) Print the results.

Implementation Notes

The computer program was written in FORTRAN 77, under the UNIX operating system, release number 4.1 by Berkeley, and should meet the 77 standard (2) in all specifics. While it was not optimized for computational efficiency, it does strive for clarity, and its structure should aid attempts at modification and improvement. All variables are passed in the call statements; there are no COMMON blocks. All of the arrays which are dependent upon the mesh are dimensioned via parameters. This minimizes storage space, but still allows for easy change in the size of the arrays when a new mesh is used.

While comments are used liberally throughout the code, the concentrated documentation is in the subroutine DOCUM. It consists of three glossaries: a glossary of routines, one of parameters, and one of variables.

The main program is routine SATE. The first routine it calls is INITAL. Subroutine INITAL creates the mesh by reading in its definition from a file, computes the areas of the elements, and zeros the global matrix.

Next, the global matrix is assembled. It is stored in four, two-dimensional array variables: M11, M12, M21, and M22. The contributions to the local matrices are computed

by five subroutines, one for each of the terms in the extremization integral. The five subroutines are STREAM, ABSORB, SCAT1, SCAT2, and SCAT3. The routine CASEDT determines the case of the orientation of the local and non-local elements, for use by the scatter routines. The contributions to the local matrices are combined into temporary matrices and saved in MAT(LOC,I,J) for later use in the calculation of the penalties. The contributions from the streaming and absorption terms are combined into one temporary matrix. The contributions from all three scatter terms are also combined into temporary matrices but there is now one temporary matrix for each non-local element in the column. This is a memory intensive approach, but it makes debugging easier and it does avoid having to recompute these matrices.

When the global matrix is completely assembled, the force vector is computed. It is just the negative product of the M12 matrix and the vector of boundary nodes. The M11 part of the global matrix is copied into the double precision array MM for two reasons. First, the implementation of IMSL on the VAX 11/780 is in double precision only. Second, the matrix sent to the IMSL routine suffers a Gauss-Siedel decomposition, and is no longer available for use in computing the global penalty.

The solution is computed by the IMSL routine LEQIF, which is a virtual memory version of the linear equation solving routine.

The penalty is computed in two ways. First, the element penalties are computed. They are useful as guides in refining the mesh. The global penalty is also computed, but for comparison purposes only, since the sum of the element penalties should be equal to the global penalty.

The results are printed on the standard output by the routine OUTPUT. It offers several options for easy tailoring of the desired information.

The last routine is DEBUG. It is a low overhead utility routine which can be used to quickly get diagnostics about the program without cluttering it up with print statements. It is no substitute for a symbolic debugger, but it is superior for problems with a lot of intermediate data.

The major variables include the previously mentioned arrays, the global matrix (M11, M12, M21, M22), and the record of the contributions to the local matrices. In addition, there are three arrays which define the mesh in use. They are CORDND, PTNODE, and AREAS. CORDND (coordinates of the nodes) contains the x_μ-coordinates of all the nodes. PTNODE (pointer to the nodes) specifies which nodes are a part of each element by pointing to CORDND. AREAS is just the area of each element.

V. Results

Local Terms

The first use of the finite element equations was to solve the benchmark problem for a blanket which consisted of a purely absorbing medium. This tested the streaming and absorption terms without the added complexity of the scattering terms. The transport equation, with absorption only, is

$$\mu \frac{\partial \phi}{\partial x} + \Sigma_t \phi = 0 \tag{266}$$

This problem, when solved analytically, is

$$\phi = c e^{-\frac{\Sigma_t}{\mu} x} \tag{267}$$

Evaluating the integration constant, C, from the boundary condition of a Lambertian source, yields

$$\phi = \mu e^{-\frac{\Sigma_t}{\mu} x} \qquad \mu > 0 \tag{268}$$

Five computer runs were done, one run for each of the five meshes. The detailed results of these runs are included in Appendix D. Some of the notable features are discussed below.

The analytic solution, for the case of $\mu = 1$, is shown in Table II, along with the nodal values and the global penalties from the five meshes.

It can be seen from Table II that, as the number of

elements in the mesh increases, the answer tends to converge to the analytic solution. The difference between Mesh 2 and Mesh 3 is not an exception, since these two meshes have the same number of elements. The only difference between them is the orientation of two adjacent elements. (Refer to the mesh maps in Appendix C.) This small change in going from Mesh 2 to Mesh 3 produced a noticeable change in both the answer and the penalty. The answer is worse in Mesh 3 than in Mesh 2 and the penalty values reflect it.

Table II

Nodal Values from Local Terms for $\mu = 1.0$

Solution	Range					Penalty	Number of Elements
	x=0.00	x=.375	x=.750	x=1.50	x=3.00		
Analytic	1.00	.687	.472	.223	.050	--	--
Mesh 1	1.00	--	.316	.116	.042	.300	9
Mesh 2	1.00	--	.413	.157	.056	.271	13
Mesh 3	1.00	--	.389	.159	.057	.278	13
Mesh 4	1.00	.637	.385	.144	.052	.264	20
Mesh 5	1.00	.657	.423	.131	.028	.247	32

The inclusion of Mesh 3 was to test the sensitivity of the answer to the orientation of the elements. Additionally, each mesh had one element with zero area as a test of the routine's ability to handle extreme cases.

For Mesh 5, for the nodes near the region where the elements are concentrated, the finite element solution is within 10 percent of the analytic solution. For a mesh of only 32 elements, this is reasonable accuracy.

In Mesh 5, the nodal values at x = 1.50 and x = 3.00 are both further from the analytic solution than for most of the earlier meshes. One possible reason for this is that the style of Mesh 5 is very different from the other meshes. In Mesh 5, the elements are concentrated in the upper right corner. This left some very large elements in the opposite corner. Another factor is that the shape of the elements near these nodes is elongated. For the finite element method, the preferred shape is a near equilateral, since this shape is better at approximating a solution with different curvatures in each direction.

In Table III, the M11 portion of the global matrix from Mesh 4 is shown. This is the part of the global matrix which couples the interior nodes to themselves. The symmetry, bandedness, and sparseness are clearly evident. A little inspection reveals that it is also positive definite. These are all the features expected. The banded structure is typical whenever an orderly numbering of the nodes is done and, like symmetry, it can be used to speed program execution or to reduce memory requirements. However, in a program which uses automatic mesh refinement, the bandedness will be disrupted with the addition of each new element. In order to

maintain the banded structure, it is necessary to utilize a re-numbering routine. A well-constructed re-numbering routine will not only maintain the banded structure, but it will also minimize its "bandwidth".

Table III

Local Term's M11 Global Matrix from Mesh 4

.73	.00	.00	.03	-.65	.00	.00	.00	.00	.00	.00	.00
.00	.99	.02	.00	.00	-.46	.00	.00	.00	.00	.00	.00
.00	.02	.89	.02	.00	-.17	-.17	.00	.00	.00	.00	.00
.03	.00	.02	.63	.03	.00	-.23	.00	.00	.00	.00	.00
-.65	.00	.00	.03	1.43	.00	.03	-.65	.00	.00	.00	.00
.00	-.46	-.17	.00	.00	1.01	.14	.00	-.30	.00	.00	.00
.00	.00	-.17	-.23	.03	.14	.87	.05	.06	-.16	.00	.00
.00	.00	.00	.00	-.65	.00	.05	1.16	.00	.06	-.30	.00
.00	.00	.00	.00	.00	-.30	.06	.00	.88	.09	.00	-.10
.00	.00	.00	.00	.00	.00	-.16	.06	.09	.77	.09	.00
.00	.00	.00	.00	.00	.00	.00	-.30	.00	.09	.81	.00
.00	.00	.00	.00	.00	.00	.00	.00	-.10	.00	.00	.29

From Appendix D, it is clear that in every mesh there are nodes whose fluxes are negative. The magnitudes are small, but the values are clearly non-physical. The nodes where these negative values occur are expected, from the analytic solution, to be small or zero, so the difference is

not great. Furthermore, small differences are not unexpected in a numerical technique, but in some applications a negative flux can be very inconvenient.

One potential solution is to use a technique called quadratic programing (10). In this technique, constraints are imposed upon the values of the nodes so they cannot go negative. Clearly, this will not improve the approximation to the extremization integral; indeed it can only worsen it, but the effect should be small, and it may be more tolerable than having negative values.

Non-Local Terms

The second use of the finite element equations was to solve the benchmark problem where the blanket had a non-zero scattering cross-section. The response of the scatter terms was the main interest.

Five computer runs were done, one run for each of the five meshes. The detailed results of these runs are included in Appendix E. Some of the notable features are discussed below.

It can be seen from Table IV that, as the number of elements in the mesh increases, the global penalty increases. These results exactly parallel the results for the local terms, right down to the inversion of Meshes 2 and 3. What really stands out, however, is the difference in the magnitudes of the penalties between the local runs and the scattering runs. In the scatter runs, the penalties are lower by

a factor of nearly three. The decrease is anticipated since the inclusion of the scatter terms smoothes the distribution of the flux. A function with a smooth distribution is better approximated by finite elements than a function whose distribution is rapidly changing.

Table IV

Penalty Values for Scattering Benchmark Problem

Solution	Penalty	Number of Elements
Mesh 1	.125	9
Mesh 2	.109	13
Mesh 3	.114	13
Mesh 4	.102	20
Mesh 5	.097	32

The penalty values for the individual elements followed about the same patterns in the scattering solution as in the absorption only solution. In particular, the relative fluctuations from element to element were about the same in both solutions.

The scattering benchmark was solved independently by another numerical technique called Ln (9). This technique was recently developed at the Air Force Institute of Technology by LCDR Kirk Mathews, U. S. Navy, who graciously ran a solution of the scattering benchmark problem. The Ln results

served as a baseline solution whose accuracy was assumed to be sufficient for the objectives of this study. The output from the Ln program is the scalar flux, not the phase space scalar flux that the routine SATE outputs, so some intermediate calculations were necessary before a comparison was possible.

The intermediate calculations were done by hand and were simple linear interpolations between the grid points of the Ln scheme and between the nodes in the finite element scheme. A comparison of the interpolations is shown in Table V. The comparison is for Mesh 5 only.

Table V

Scalar Flux for Scattering Benchmark Problem

Solution	Range					Penalty	Number of Elements
	x=0.00	x=.375	x=.750	x=1.50	x=3.00		
Ln	.296	.191	.129	.064	.016	--	--
Mesh 5	.285	.235	.182	.091	.036	.097	32

The finite element solution disagrees with the Ln solution by about 35 percent. At x = 0, the value of the flux is driven by the boundary conditions; therefore, the difference at this point is a rough measure of the accuracy of the intermediate calculations. The apparent uncertainty in the intermediate calculations is clearly not enough to

explain the larger differences over the mesh as a whole.

The finite element solution is larger at every point, which indicates a systematic error. It seems that the scatter terms are over-active; that is, their contribution to the solution is disproportionate to the contribution time from the local terms.

Before addressing possible causes of the difference, the global matrix should be inspected. The M11 portion of the global matrix from Mesh 4 is shown in Table VI. Mesh 4 is shown instead of Mesh 5 for several reasons. First and foremost, Mesh 4 possesses all of the significant features from any of the global matrices from the scattering runs. Additionally, the global matrix shown in Table III for the local terms was also taken from Mesh 4, and so a direct comparison is possible. Lastly, the global matrix from Mesh 4 is the largest from the set of five meshes that will conveniently fit on a page.

The scattering global matrix displays a typical banded structure and it is positive definite, both expected features. Also, it is much less sparse than the global matrix from the absorption only runs. This is because of the non-localness of the scattering terms. The decrease in the sparseness is also an expected feature. The critical feature, however, is the lack of symmetry. With the two digits of precision used in outputing the matrix, it is clear that many of the cross-diagonal terms are not equal. While the

differences are small, this lack of symmetry is inconsistent with a self-adjoint equation. Clearly, the local matrices for the scatter terms are being computed incorrectly.

Table VI

Scatter Term's M11 Global Matrix from Mesh 4

.70	-.02	-.02	.00	-.65	.00	.00	.00	.00	.00	.00	.00
-.02	.98	.00	-.02	-.00	-.47	.00	.00	.00	.00	.00	.00
-.02	.01	.86	-.03	-.03	-.18	-.18	.01	.00	.00	.00	.00
.01	-.02	-.03	.57	.01	-.01	-.24	.00	.00	.00	.00	.00
-.66	-.00	-.03	.00	1.39	-.03	-.01	-.65	.00	.00	.00	.00
.00	-.46	-.18	-.01	-.03	.98	.07	-.02	-.31	-.00	.00	.00
.00	.00	-.18	-.24	-.01	.08	.69	-.03	.01	-.20	.00	.00
.00	.00	.01	.00	-.66	-.02	-.03	1.10	-.05	.01	-.31	.00
.00	.00	.00	.00	.00	-.31	.01	-.05	.76	-.06	-.10	-.12
.00	.00	.00	.00	.00	.00	-.20	.01	-.06	.55	-.02	-.02
.00	.00	.00	.00	.00	.00	-.00	-.31	-.08	-.03	.71	-.03
.00	.00	.00	.00	.00	.00	.00	.00	-.12	-.01	-.03	.27

There are at least three possible sources of the error. First, the local matrices may not have been derived correctly, second, the finite element equations may not have been correctly coded into the computer routines, and third, the calculation of the self-adjoint operator may be incorrect. If either of the first two possibilities is the source of

the error, then correcting them should restore symmetry to the global matrix.

The local matrices from the second and third scatter terms are not symmetric (see the debug output in Appendix F). This was apparent at the time of the derivation and was not of concern for two reasons. First, the self-adjoint property the usual terms found in self-adjoint equations. Recall that the scatter terms in the extremization integral are to be integrated over the entire range of μ'. In addition, in this derivation they are evaluated over one non-local element at a time. For these reasons, the contributions to the local matrices from the scatter terms are not expected to be symmetric.

What should happen, however, is that the combined contributions to the local matrix from all the scatter terms be symmetric. Obviously, this did not happen since the global matrix was not symmetric.

The possibility remains of coding errors in the computer program. Several errors were found in the final days of this study and there are no guarantees that all the bugs were eradicated.

In the results for the local terms, the fluxes near the right boundary did not converge consistently as the mesh was refined. It was attributed to the relative coarseness of the meshes in that region. It is possible, however, that another effect is in operation.

When the self-adjoint operator was derived, it was computed from $\mathcal{L}^*\mathcal{L}$. If instead, it is derived from the extremization integral (Equation 53), then a surface term appears. The surface term does not change the equations of motion, but it does prescribe that the conventional transport equation be satified on the free boundaries, a proper result. This term is missing in the derivation done in Chapter II and, therefore, there may be an incorrect boundary condition in effect. This would put strains on the solution and would seriously perturb it near the free boundaries. It would be useful to have a uniform mesh in investigating this possibility.

In the scattering runs, not a single node was found to have negative values, not for any of the meshes. This is reasonable, since the process of scatter tends to smooth and spread the distribution of the flux.

The real-time run time for the benchmark problem with scatter, using Mesh 5, was 7 seconds. This was on a VAX 11/780 with no other major programs running. The FORTRAN compiler on this particular machine is very inefficient, however, since the FORTRAN is translated into the programing language C before compilation. In addition, this implementation of the finite element equations was not optimized for computational efficiency.

VI. Conclusion and Recommendations

Conclusion

This study accomplished three major tasks toward a general solution of the transport equation using finite elements: the recasting of the transport equation into a self-adjoint form, the derivation of the finite element equations for the one-dimensional case with isotropic scatter, and the implementation of these equations in a computer code. While the solution has the expected features, some questions remain about residual errors in the scatter terms.

Several features of the finite element solution stand out. The integer nature of the cores of the local matrices can be used to speed implementation by coding the subroutines which utilize them in assembly language. The difficulty in deriving the scatter terms may become an insurmountable obstacle when non-isotropic scatter is considered. The usual approach used for non-isotropic scatter is to expand the flux in Legendre polynomials. This will raise the order of the terms in the scatter integrals and significantly increase the difficulty of evaluating them. Since the local matrices are needed for the penalty calculation, it is necessary either to save them or to recompute them. The former is memory-intensive; the latter increases the execution time. The process

of making a mesh by hand is time-consuming and error-prone. The lack of automatic mesh refinement means that large meshes are not readily attainable. Lastly, the columnar scheme reduces the flexibility of the finite element method, and thus increases the required number of numerical computations needed for a given level of accuracy. It does not seem to be a severe limitation, but specialized applications might find it cumbersome.

There are many benefits to be gained from using finite elements to solve the transport equation: the ability to concentrate the numerical calculations in regions where the solution is rapidly changing, the symmetry and sparseness of the global matrix, and the speed of the calculations involved in the finite element approximation. This study has charted a path toward the use of the finite element method that hopefully will be of benefit to those who may choose to advance the technique even further.

Recommendations

The three potential sources of the error in the solution should be investigated first. Once a working program is achieved, the way is then clear to improve and expand upon the technique explored in this study.

The basic technique explored in this study could be improved in two ways. First, the amount of mathematical

manipulations needed to derive the finite element equations for the scatter terms should be reduced. This can be done by giving some consideration to the development of theorems or formulas, such as the triangular integration formula. For the Legendre polynomial expansion of the flux, these formulas may indeed by a necessity. Additionally, the use of such theorems should increase the confidence in the results of the derivation.

Second, alternatives to the columnar scheme should be considered. This does not seem to be a high priority, but the current scheme does somewhat reduce the relative advantage of the finite element method because it reduces its ability to concentrate numerical calculations in selected regions.

There are a few further investigations that can be done to increase the range of problems which can be handled. Two immediate needs are the inclusion of the finite element equations for sources, and the derivation of the finite element equations for non-isotropic scatter. This method should be considered incompletely tested until an automatic mesh refinement scheme is developed. With this addition, this method could be tested quantitatively for speed and accuracy.

The ultimate goal would be the extension to two dimensions. Occasionally, a numerical technique works well in one dimension, but, for one reason or another, cannot be

implemented in two, much less three. The extension to two dimensions would make this technique available for use in the solving of true engineering problems.

Bibliography

1. Ayres, Frank, Jr. Schaum's Outline of Theory and Problems of Matrices. New York: McGraw-Hill Book Company, 1962.

2. Balfour, A. and D. H. Marwick. Programming in Standard Fortran 77. New York: Elsevier North Holland, Inc., 1981.

3. Desai, ChandraKant S. and John F. Abel. Introduction to the Finite Element Method; A Numerical Method for Engineering Analysis. New York: Van Nostrand Reinheld, 1972.

4. Duderstadt, James J. and William R. Martin. Transport Theory. New York: John Wiley & Sons, Inc., 1979.

5. Ewing, George M. Calculus of Variations with Applications. New York: W. W. Norton & Company, Inc., 1969.

6. Fix, George J. and Gilbert Strang. An Analysis of the Finite Element Method. New Jersey: Prentice-Hall, Inc., 1973.

7. Huebner, Kenneth H. The Finite Element Method for Engineers. New York: John Wiley & Sons, Inc., 1975.

8. Lewins, Jeffery. Importance: The Adjoint Function. Oxford: Pergamon Press, Ltd., 1965.

9. Mathews, LCDR Kirk A. Discrete Elements Method of Neutral Particle Transport. Dissertation (to be published) GNE/EN/83D, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1983.

10. Shankland, Donn G. Quadratic Programming Using Generalized Inverses. AFIT-TR-75-2.. Wright-Patterson AFB OH: Air Force Institute of Technology, May 1975.

Appendix A

Evaluation of Five Element Integrals

This appendix contains the derivations of the five intervals presented in Table I in Chapter III.

The first integral is

$$\int L_i \, dA = \frac{A}{3} \tag{269}$$

where A is the area of the local element. Expressing all the integrals indexed by i as a vector, yields

$$\int \underline{L} \, dA = \begin{bmatrix} A/3 \\ A/3 \\ A/3 \end{bmatrix} \tag{270}$$

or, for compatibility with later integrals, as a matrix vector product, and with different scaling,

$$\int \underline{L} \, dA = \frac{A}{60} \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{271}$$

Defining

$$\underline{\underline{m}}_a = \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix} \tag{272}$$

This becomes

$$\int \underline{L} \, dA = \frac{A}{60} \underline{\underline{m}}_a \underline{I} \tag{273}$$

where $\underline{I}$ is the unit vector.

The second integral becomes

$$\int L_i x \, dA = \int L_i (x_1 L_1 + x_2 L_2 + x_3 L_3) \, dA \tag{274}$$

which, in vector form, is

$$\int \underline{L} x \, dA = \int \begin{bmatrix} L_1^2 & L_1 L_2 & L_1 L_3 \\ L_2 L_1 & L_2^2 & L_2 L_3 \\ L_3 L_1 & L_3 L_2 & L_3^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} dA \tag{275}$$

which, from the triangular integration formula with scaling, is simply

$$\int \underline{L} x \, dA = \begin{bmatrix} 10 & 5 & 5 \\ 5 & 10 & 5 \\ 5 & 5 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{276}$$

which, from previous definitions, is

$$\int \underline{L} x \, dA = \frac{A}{60} \underline{\underline{m}}_a \underline{x} \tag{277}$$

where

$$\underline{x} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{278}$$

The third integral, by symmetry with the second, is

$$\int \underline{L} \mu \, dA = \frac{A}{60} \underline{\underline{m}}_a \underline{\mu} \tag{279}$$

where

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{280}$$

The fourth integral becomes

$$\int L_i x^2 dA = \int L_i (x_1 L_1 + x_2 L_2 + x_3 L_3)^2 dA \tag{281}$$

or

$$\int L_i x^2 dA = \int L_i \sum_j^3 x_j L_j \sum_k^3 x_k L_k \, dA \tag{282}$$

and

$$\int L_i x^2 dA = \sum_{jk}^3 x_j \int L_i L_j L_k \, dA \, x_k \tag{283}$$

from the triangular integration formula

$$\int L_i L_j L_k \, dA = 6 \frac{A}{60} \qquad i = j = k \tag{284}$$

$$\int L_i L_j L_k \, dA = 2 \frac{A}{60} \qquad i = j, \; i = k, \; j = k \tag{285}$$

$$\int L_i L_j L_k \, dA = 1 \frac{A}{60} \qquad i \neq j \neq k \tag{286}$$

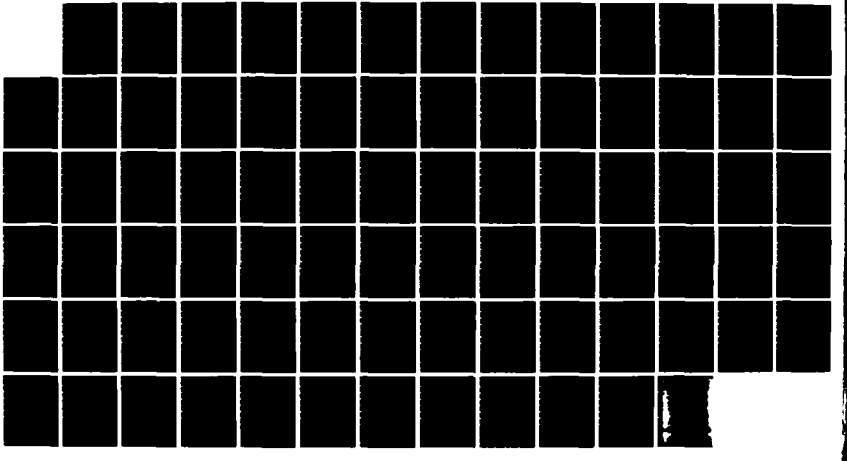
In matrix form, where the matrix indices are over j and k, not over i,

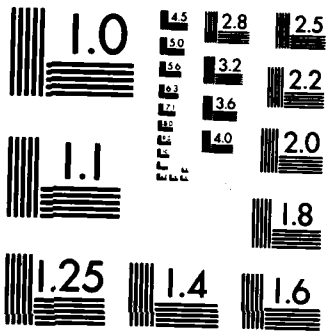
$$\int L_i x^2 dA = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \int \begin{bmatrix} L_1^2 & L_1 L_2 & L_1 L_3 \\ L_2 L_1 & L_2^2 & L_2 L_3 \\ L_3 L_1 & L_3 L_2 & L_3^2 \end{bmatrix} dA \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{287}$$

or

AD-A141 225 FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION(U) AIR FORCE INST OF TECH WRIGHT-PATTESSON AFB OH SCHOOL OF ENGI.. A D GOFF 2/2

UNCLASSIFIED MAR 84 AFIT/GNE/PH/84M-4 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$\int L_i x^2 dA = \frac{A}{60} \tilde{\underline{x}} \underline{\underline{m}}_i \underline{x} \tag{288}$$

where

$$\underline{\underline{m}}_i = \int\int \begin{bmatrix} L_1^2 & L_1 L_2 & L_1 L_3 \\ L_2 L_1 & L_2^2 & L_2 L_3 \\ L_3 L_1 & L_3 L_2 & L_3^2 \end{bmatrix} dA \tag{289}$$

for $i = 1, 2, 3$, $\underline{\underline{m}}_i$ becomes

$$\underline{\underline{m}}_1 = \begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \tag{290}$$

$$\underline{\underline{m}}_2 = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & 2 & 2 \end{bmatrix} \tag{291}$$

$$\underline{\underline{m}}_3 = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 6 \end{bmatrix} \tag{292}$$

Note that

$$\underline{\underline{m}}_1 + \underline{\underline{m}}_2 + \underline{\underline{m}}_3 = \underline{\underline{m}}_a \tag{293}$$

Including all three matrices, the fourth integral is

$$\int \underline{L}\, x^2 dA = \frac{A}{60} \tilde{\underline{x}} \begin{bmatrix} \underline{\underline{m}}_1 \\ \underline{\underline{m}}_2 \\ \underline{\underline{m}}_3 \end{bmatrix} \underline{x} \tag{294}$$

Explicitly bringing out the columnar nature of the integral

$$\int \underline{L}\, x^2 dA = \frac{A}{60}\left\{\begin{array}{l} [x_1\ x_2\ x_3]\begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}\underline{X} \\ [x_1\ x_2\ x_3]\begin{bmatrix} 2 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & 2 & 2 \end{bmatrix}\underline{X} \\ [x_1\ x_2\ x_3]\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 6 \end{bmatrix}\underline{X} \end{array}\right\} \tag{295}$$

where the first row corresponds to $i = 1$, the second row corresponds to $i = 2$, etc. Changing the form of the expression by multiplying the first vector with the matrix, then rearranging and combining terms, yields

$$\int \underline{L}\, x^2 dA = \frac{A}{60}\left\{x_1\begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} + x_2\begin{bmatrix} 2 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & 2 & 2 \end{bmatrix} + x_3\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 6 \end{bmatrix}\right\}\underline{X} \tag{296}$$

but since $x_2 = x_3$, then

$$\int \underline{L}\, x^2 dA = \frac{A}{60}\left\{x_1\begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} + x_2\begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix}\right\}\underline{X} \tag{297}$$

Define

$$\underline{\underline{M}}_b = \begin{bmatrix} 4 & 3 & 3 \\ 3 & 8 & 4 \\ 3 & 4 & 8 \end{bmatrix} = \underline{\underline{M}}_2 + \underline{\underline{M}}_3 \tag{298}$$

Thus,

$$\int_L x^2 dA = \frac{A}{60} \{ x_1 \underline{m}_1 + x_2 \underline{m}_b \} \underline{\alpha} \tag{299}$$

The fifth term, by symmetry with the fourth, is simply

$$\int_L x \underline{u} dA = \frac{A}{60} \{ x_1 \underline{m}_1 + x_2 \underline{m}_b \} \underline{u} \tag{300}$$

Appendix B

FORTRAN Source Code

```
      PROGRAM SATE
*****************************************************************************
*                                                                           *
*   SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION BY FINITE ELEMENTS        *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

C ** Indexes.
      INTEGER  I,J,K,TRIANG,NONLOC,LOCMAT

C ** Output flags.
      LOGICAL  FMESH,FNODES,FELEM,FMTRXI,FMTRXB,FFORCE

C ** Mesh parameters.
      INTEGER  MINOD,MBNOD,MNTRIA,MLOCMT,SIZE

C ** Maximum dimensions of the global matrix and number of elements.
      PARAMETER (       MINOD   =   64          ,
     +                  MBNOD   =   16          ,
     +                  MNTRIA  =   64          ,
     +                  MLOCMT  =  512          ,
     +                  SIZE    =  MINOD+MBNOD   )

C ** Size data structures.
      INTEGER  PTNODE(MNTRIA,4),COLUMN(32,2)
      REAL     CORDND(MINOD+MBNOD,2),PSI(MBNOD),PHI(MINOD)
      REAL     AREAS(MNTRIA),PENLTY(MNTRIA)
      REAL     M11(MINOD,MINOD),M12(MINOD,MBNOD)
      REAL     M21(MBNOD,MINOD),M22(MBNOD,MBNOD)
      REAL     MAT(MLOCMT,3,3)

C ** Mesh variables.
      INTEGER  INTNOD,BNDNOD,NTRIAN
      REAL     RANGE,SIGMAT,SIGMAS

C ** Computations.
      INTEGER  IER
      REAL     AREA,M(3,3)
      DOUBLE PRECISION MM(MINOD,MINOD),B(MINOD),WK(3*MINOD)
      LOGICAL  BUG

*****************************************************************************

C ** Initialize the element characteristics.
      CALL INITAL(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,
     +          PTNODE,COLUMN,CORDND,PHI,PSI,AREAS,PENLTY,
     +          M11,M12,M21,M22,RANGE,SIGMAT,SIGMAS,B)

C ** Setup the output flags.
      FMESH  = .TRUE.
      FNODES = .TRUE.
```

```
      FELEM  = .TRUE.
      FMTRXI = .TRUE.
      FMTRXB = .TRUE.
      FFORCE = .TRUE.

*****************************************************************************

C ** Assemble the matrix.
      LOCMAT = 1
      DO 300 TRIANG=1,NTRIAN
        AREA = AREAS(TRIANG)
        DO 310 I=1,3
          DO 310 J=1,3
            MAT(LOCMAT,I,J) = 0.
  310   CONTINUE

        CALL STREAM(TRIANG,AREA,MNTRIA,SIZE,PTNODE,CORDND,M)
        CALL ASEMBL(TRIANG,TRIANG,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                            PTNODE,M,M11,M12,M21,M22)
        DO 320 I=1,3
          DO 320 J=1,3
  320       MAT(LOCMAT,I,J) = MAT(LOCMAT,I,J) + M(I,J)

        CALL ABSORB(TRIANG,AREA,SIGMAT,M)
        CALL ASEMBL(TRIANG,TRIANG,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                            PTNODE,M,M11,M12,M21,M22)
        DO 330 I=1,3
          DO 330 J=1,3
  330       MAT(LOCMAT,I,J) = MAT(LOCMAT,I,J) + M(I,J)

        LOCMAT = LOCMAT +1
        NONLOC = COLUMN(PTNODE(TRIANG,4),1)
        DO 355 K=1,COLUMN(PTNODE(TRIANG,4),2)

          CALL SCAT1(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                            SIGMAT,SIGMAS,M)
          CALL ASEMBL(TRIANG,NONLOC,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                            PTNODE,M,M11,M12,M21,M22)
          DO 340 I=1,3
            DO 340 J=1,3
  340         MAT(LOCMAT,I,J) = MAT(LOCMAT,I,J) + M(I,J)

          CALL SCAT2(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                            SIGMAS,M)
          CALL ASEMBL(TRIANG,NONLOC,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                            PTNODE,M,M11,M12,M21,M22)
          DO 350 I=1,3
            DO 350 J=1,3
  350         MAT(LOCMAT,I,J) = MAT(LOCMAT,I,J) + M(I,J)

          CALL SCAT3(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                            SIGMAS,M)
```

```
            CALL ASEMBL(TRIANG,NONLOC,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                                PTNODE,M,M11,M12,M21,M22)
            DO 360 I=1,3
              DO 360 J=1,3
  360           MAT(LOCMAT,I,J) = MAT(LOCMAT,I,J) + M(I,J)

            NONLOC = NONLOC +1
            LOCMAT = LOCMAT +1
  355     CONTINUE

C -- Record the assembling of the matrix.
        CALL DEBUG('SATE  ',BUG)
        IF(BUG)  THEN
          PRINT *, 'SATE***          TRIANG= ',TRIANG
          CALL OUTPUT(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE,
     +                CORDND,PHI,PSI,AREAS,PENLTY,M11,M12,M21,M22,B,
     +                .FALSE.,.FALSE.,.FALSE.,FMTRXI,.FALSE.,.FALSE.)
        ENDIF

  300 CONTINUE

C ** Compute the force vector.
        DO 400 I=1,INTNOD
          B(I) = 0.
          DO 400 J=1,BNDNOD
            B(I) = B(I) - M12(I,J)*PSI(J)
  400 CONTINUE

C ** Record the pre-solution status.
        CALL DEBUG('SATE  ',BUG)
        IF(BUG)  THEN
          CALL OUTPUT(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE,
     +                CORDND,PHI,PSI,AREAS,PENLTY,M11,M12,M21,M22,B,
     +                FMESH,FNODES,FELEM,FMTRXI,FMTRXB,FFORCE)
        ENDIF

C ** Convert to double precision for IMSL.  Also, advert decomposition.
        DO 500 I=1,INTNOD
          DO 500 J=1,INTNOD
            MM(I,J) = M11(I,J)
  500 CONTINUE

C ** Solve the SATE matrix equation.
        CALL LEQIF(MM,MINOD,INTNOD,INTNOD,B,MINOD,1,0,WK,IER)
        WRITE(*,'(/'LEQIF'/'IER= ',I7,/)')  IER

C ** Convert the LEQIF returned nodal values from B to PHI.
        DO 600 I=1,INTNOD
          PHI(I) = B(I)
  600 CONTINUE

C ** Compute the penalty of each element.
```

```
      CALL PENAL(NTRIAN,MNTRIA,MINOD,MBNOD,MLOCMT,INTNOD,BNDNOD,PTNODE,
     +           COLUMN,PHI,PSI,MAT,M11,M12,M21,M22,PENLTY)

C ** Output the results.
        CALL OUTPUT(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE,
     +              CORDND,PHI,PSI,AREAS,PENLTY,M11,M12,M21,M22,B,
     +              FMESH,FNODES,FELEM,FMTRXI,FMTRXB,FFORCE)

*******************************************************************************
      STOP
      END
```

```
      SUBROUTINE DOCUM
C*****************************************************************************
C*                                                                           *
C*                            DOCUMENTATION                                  *
C*                                                                           *
C*****************************************************************************

C*****************************************************************************
C*                                                                           *
C*                            1Lt Allan Goff                                 *
C*                                                                           *
C*                          December 31, 1983                                *
C*                                                                           *
C*      This program solves the self-adjoint transport equation in           *
C*  a finite element scheme.                                                 *
C*                                                                           *
C*****************************************************************************

C*****************************************************************************
C*                          SUBROUTINE GLOSSARY                              *
C*                                                                           *
C*      SATE    - The calling program.                                       *
C*      DOCUM   - The documentation routine.                                 *
C*      INITAL  - Initializes data structures and element constants.         *
C*      STREAM  - Computes the contribution from the streaming term.         *
C*      ABSORB  - Computes the contribution from the absorption term.        *
C*      SCAT1   - Computes the contribution from the 1st scatter term.       *
C*      SCAT2   - Computes the contribution from the 2nd scatter term.       *
C*      SCAT3   - Computes the contribution from the 3rd scatter term.       *
C*      CASEDT  - Determines symmetry or assymmetry between elements.       *
C*      ASEMBL  - Assembles the global matrix from the local matrices.       *
C*      PENAL   - Computes the minimization integral for each element.       *
C*      OUTPUT  - Prints selected intermediate and final data.               *
C*      DEBUG   - A low over-head debugging routine.                         *
C*                                                                           *
C*****************************************************************************

C*****************************************************************************
C*                          PARAMETER GLOSSARY                               *
C*                                                                           *
C*      MNTRIA  - Maximum number of elements.                                *
C*      MINOD   - Maximum number of interior nodes.                          *
C*      MBNOD   - Maximum number of boundary nodes.                          *
C*      MLOCMT  - Maximum number of local matrices.                          *
C*      PI      - Ratio of diameter of a circle to its circumference.        *
C*                                                                           *
C*****************************************************************************

C*****************************************************************************
C*                          VARIABLE GLOSSARY                                *
C*                                                                           *
C*  NTRIAN -  Number of elements in this mesh.                               *
```

```
*  INTNOD -  Number of interior nodes in this mesh.                    *
*  BNDNOD -  Number of boundary nodes in this mesh.                    *
*  LOCMAT -  Number of local matrices.                                 *
*  LOC    -  Number of local matrices.                                 *
*  SIGMAT      - The total macroscropic cross section.                 *
*  SIGMAS      - The absorption macroscropic cross section.            *
*  RANGE       - Range of the x-axis in units of sigmat.               *
*  I,J,K  -  Indexes for general use.                                  *
*  II     -  The row index on the global matrix.                       *
*  JJ     -  The column index on the global matrix.                    *
*  TRIANG -  Integer specifying the current local triangle.            *
*  NONLOC -  Integer specifying the current non-local triangle.        *
*  X(3)   -  X-coordinate of the ith vertex of the current element.    *
*  U(3)   -  Mu-coordinate of the ith vertex of the current element.   *
*  A(3)   -  X coefficient in the linear finite element approximation. *
*  B(3)   -  Mu coefficient in the linear finite element approximation.*
*  C(3)   -  Constant coefficient in the linear finite element approx. *
*  AREA   -  Area of the current triangle.                             *
*  AREAS()-  Areas of the triangles.                                   *
*  PHI()  -  Vector of the global interior nodal values.               *
*  PSI()  -  Vector of the global boundary nodal values.               *
*  PHILOC(3) -  Vector of the local interior nodal values.             *
*  PHIFRM(3) -  Vector of the non-local interior nodal values.         *
*  M11(,) -  The global matrix, interior node section.                 *
*  M12(,) -  The global matrix, boundary node section, off diagonal.   *
*  M21(,) -  The global matrix, boundary node section, off diagonal.   *
*  M22(,) -  The global matrix, boundary node section, on diagonal.    *
*  MM(,)  -  The global matrix, interior node section, double-p copy.  *
*  M(3,3) -  The local matrix.                                         *
*  MAT(,,)-  Record of local matrices.  Used to compute the penalty.   *
*  B()    -  Force vector of the SATE matrix equation.                 *
*  M1(,)  -  Integer matrix from scatter terms, X(1)-coefficient.      *
*  MA(,)  -  Integer matrix from scatter terms, X(1)&X(2)-coefficient. *
*  MB(,)  -  Integer matrix from scatter terms, constant coefficient.  *
*  MC(,)  -  Integer matrix from first scatter term, symmetric cases.  *
*  MD(,)  -  Integer matrix from first scatter terms, antisymmetric '. *
*  PTNODE(,3)-  Pointers to the global nodes from the element vertexes.*
*  CORDND(,2)-  Coordintes of the global nodes.                        *
*  COLUMN(,2)-  Number of elements in each column.                     *
*  SIZE   -  The total number of global nodes.                         *
*  P      -  Common factors in the local matrix.                       *
*  PP     -  Common factors in the local matrix.                       *
*  PEN    -  The penalty value from the working element.               *
*  PENLTY()  - The penalty values for all the elements.                *
*  WK()   -  The 3*PINOD working space for the IMSL routine LEQIF.     *
*  CASE   -  Indicates symmetric or assymetric case between elements.  *
*  BUG    -  Debugging flag used to select the debugging routine.      *
*  FMESH  -  Output option flag for mesh definition.                   *
*  FNODES -  Output option flag for the nodal vectors.                 *
*  FELEM  -  Output option flag for element charactoristics.           *
*  FMTRXI -  Output option flag for global matrix.                     *
*  FMTRXB -  Output option flag for global matrix.                     *
```

```
*  FFORCE -  Output option flag for force vector.                              *
*                                                                              *
*                                                                              *
********************************************************************************
      RETURN
      END
```

```
      SUBROUTINE INITAL(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,
     +           PTNODE,COLUMN,CORDND,PHI,PSI,AREAS,PENLTY,
     +           M11,M12,M21,M22,RANGE,SIGMAT,SIGMAS,B)
*****************************************************************************
*                                                                           *
*                        INITIAL COMPUTATIONAL DATA                         *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

C ** Indexes.
      INTEGER  I,J,K,TRIANG,NODE

C ** Passed variables.
      INTEGER  MNTRIA,MINOD,MBNOD

      INTEGER  NTRIAN,INTNOD,BNDNOD,NCOL
      INTEGER  PTNODE(MNTRIA,4),COLUMN(32,2)
      REAL     CORDND(MINOD+MBNOD,2)
      REAL     PHI(MINOD),PSI(MBNOD)
      REAL     AREAS(MNTRIA),PENLTY(MNTRIA)
      REAL     M11(MINOD,MINOD),M12(MINOD,MBNOD)
      REAL     M21(MBNOD,MINOD),M22(MBNOD,MBNOD)
      REAL     RANGE,SIGMAT,SIGMAS
      DOUBLE PRECISION B(MINOD)

C ** Computations.
      REAL       AREA,U(3),X(3)
      CHARACTER  TRASH*16
      LOGICAL    BUG

*****************************************************************************

C ** Read in the mesh definition.
      OPEN(3,FILE='Data/mesh')
      REWIND 3

        READ(3,'(A16)')  TRASH
        READ(3,'(4(1X,I7))')  NTRIAN,INTNOD,BNDNOD,NCOL
        READ(3,'(1X)')

        READ(3,'(A16)')  TRASH
        READ(3,'(3(1X,F7.3))')  RANGE,SIGMAT,SIGMAS
          RANGE = RANGE*SIGMAT
        READ(3,'(1X)')

        READ(3,'(A16)')  TRASH
        DO 50 I=1,NTRIAN
          READ(3,'(1X,I7,8X,4(1X,I7))')  TRIANG,(PTNODE(I,J),J=1,4)
   50   CONTINUE
        READ(3,'(1X)')
```

```
      READ(3,'(A16)')  TRASH
      DO 60 I=1,NCOL
        READ(3,'(3(1X,I7,8X))')  TRIANG,(COLUMN(I,J),J=1,2)
60    CONTINUE
      READ(3,'(1X)')

      READ(3,'(A16)')  TRASH
      DO 70 I=1,INTNOD+BNDNOD
        READ(3,'(1X,I7,8X,2(1X,F7.3))')  NODE,(CORDND(I,J),J=1,2)
        CORDND(I,1) = CORDND(I,1)*RANGE
70    CONTINUE
      READ(3,'(1X)')

      READ(3,'(A16)')  TRASH
      DO 80 I=1,BNDNOD
        READ(3,'(1X,I7,8X,1(1X,F7.3))')  NODE,PSI(I)
80    CONTINUE
      READ(3,'(1X)')

    CLOSE(3)

C ** Compute the areas of each of the triangular elements.
      DO 100 TRIANG=1,NTRIAN
        U(3) = CORDND(PTNODE(TRIANG,3),2)
        U(2) = CORDND(PTNODE(TRIANG,2),2)
        X(2) = CORDND(PTNODE(TRIANG,2),1)
        X(1) = CORDND(PTNODE(TRIANG,1),1)
        AREA = ABS( .5*( U(3)-U(2) )*( X(2)-X(1) ) )
        IF(AREA .LT. 1.0E-15)  AREA = 1.0E-15
        AREAS(TRIANG) = AREA
        PENLTY(TRIANG) = 0.
100   CONTINUE

C ** Zero the global matrix and the nodal vector.
      DO 200 I=1,INTNOD
        PHI(I) = 0.
        DO 210 J=1,INTNOD
210       M11(I,J) = 0.
        DO 220 K=1,BNDNOD
          M12(I,K) = 0.
220       M21(K,I) = 0.
200   CONTINUE
      DO 230 I=1,BNDNOD
        DO 230 J=1,BNDNOD
          M22(I,J) = 0.
230   CONTINUE

*****************************************************************************
      CALL DEBUG('INITAL',BUG)
      IF(BUG)  THEN
        PRINT *, 'INITAL***'
        CALL OUTPUT(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE,
```

```
     +              CORDND,PHI,PSI,AREAS,PENLTY,M11,M12,M21,M22,B,
     +              .TRUE.,.TRUE.,.TRUE.,.TRUE.,.TRUE.,.TRUE.)
        WRITE(*,'(32(1X,I4,2X,I4/))')    ((COLUMN(I,J),J=1,2),I=1,NCOL)
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ASEMBL(TRIANG,NONLOC,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD,
     +                                        PTNODE,M,M11,M12,M21,M22)
*****************************************************************************
*                                                                           *
*                          Assemble the Global Matrix                       *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,NONLOC,MNTRIA,MINOD,MBNOD,INTNOD,BNDNOD
      INTEGER  PTNODE(MNTRIA,4)
      REAL     M(3,3)

      REAL     M11(MINOD,MINOD),M12(MINOD,MBNOD)
      REAL     M21(MBNOD,MINOD),M22(MBNOD,MBNOD)

      INTEGER  I,J,II,JJ
      LOGICAL  BUG

*****************************************************************************

C ** Relocate the local matrix elements into the global matrix.
      DO 100 I=1,3
        DO 100 J=1,3
          II = PTNODE(TRIANG,I)
          JJ = PTNODE(NONLOC,J)
          IF( (II .LE. INTNOD) .AND. (JJ .LE. INTNOD) )  THEN
            M11(II,JJ) = M11(II,JJ) + M(I,J)
          ELSE IF( (II .LE. INTNOD) .AND. (JJ .GT. INTNOD) )  THEN
            M12(II,JJ-INTNOD) = M12(II,JJ-INTNOD) + M(I,J)
          ELSE IF( (II .GT. INTNOD) .AND. (JJ .LE. INTNOD) )  THEN
            M21(II-INTNOD,JJ) = M21(II-INTNOD,JJ) + M(I,J)
          ELSE IF( (II .GT. INTNOD) .AND. (JJ .GT. INTNOD) )  THEN
            M22(II-INTNOD,JJ-INTNOD) = M22(II-INTNOD,JJ-INTNOD)
     +                                              +M(I,J)
          ELSE
            PRINT *, 'Impossible option in ASEMBL.'
          ENDIF
  100 CONTINUE

*****************************************************************************
      CALL DEBUG('ASEMBL',BUG)
      IF(BUG)  THEN
        PRINT *, 'ASEMBL***'
        DO 8800 II=1,INTNOD+BNDNOD
          IF(II .LE. INTNOD)  THEN
            WRITE(*,6210) (M11(II,J),J=1,INTNOD),(M12(II,J),J=1,BNDNOD)
 6210       FORMAT(1X,'|',20(1X,F6.3))
            WRITE(*,6220)
 6220       FORMAT(1X,'|')
          ELSE
            I = II - INTNOD
```

```
                WRITE(*,6210) (M21(I,J),J=1,INTNOD), (M22(I,J),J=1,BNDNOD)
                WRITE(*,6220)
              ENDIF
 8800    CONTINUE
       ENDIF

       RETURN
       END
```

```
      SUBROUTINE STREAM(TRIANG,AREA,MNTRIA,SIZE,PTNODE,CORDND,M)
*****************************************************************************
*                                                                           *
*                              STREAMING TERM                               *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,MNTRIA,SIZE,PTNODE(MNTRIA,4)
      REAL     AREA,CORDND(SIZE,2)

      INTEGER  I,J
      REAL     U(3),A(3),P,PP,M(3,3)
      LOGICAL  BUG

*****************************************************************************
C ** Compute the mu-coordinates of each vertex.
      U(1) = CORDND(PTNODE(TRIANG,1),2)
      U(2) = CORDND(PTNODE(TRIANG,2),2)
      U(3) = CORDND(PTNODE(TRIANG,3),2)

C ** Compute the common factors in the local matrix.
      P  = U(1)*(U(1)+U(2)) + U(2)*(U(2)+U(3)) + U(3)*(U(3)+U(1))
      PP = SQRT(P/(24.*AREA))

C ** Compute the x-coordinate coefficient for each vertex.
      A(1) = PP*(U(2)-U(3))
      A(2) = PP*(U(3)-U(1))
      A(3) = PP*(U(1)-U(2))

C ** Compute the matrix elements.
      DO 100 I=1,3
        DO 100 J=1,I
          M(I,J) = A(I)*A(J)
          M(J,I) = M(I,J)
  100 CONTINUE

*****************************************************************************
      CALL DEBUG('STREAM',BUG)
      IF(BUG)  THEN
        PRINT *, 'STREAM***      TRIANG=',TRIANG
        DO 8800 I=1,3
          WRITE(*,8600) (M(I,J),J=1,3),U(I),A(I),P,PP
 8600     FORMAT(1X,3(F7.3,1X),4X,4(F7.3,3X))
 8800   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ABSORB(TRIANG,AREA,SIGMAT,M)
*************************************************************************
*                                                                       *
*                            ABSORPTION TERM                            *
*                                                                       *
*************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG
      REAL     AREA,SIGMAT

      REAL     M(3,3)

      INTEGER  I,J
      REAL     PP
      LOGICAL  BUG

*************************************************************************

C ** Compute the common factors in the local matrix elements.
      PP = SIGMAT*SIGMAT*AREA/12.

C ** Fill in the local matrix.
      M(1,1) = PP*2.
      M(1,2) =   PP*1.
      M(1,3) =     PP*1.

      M(2,1) = PP*1.
      M(2,2) =   PP*2.
      M(2,3) =     PP*1.

      M(3,1) = PP*1.
      M(3,2) =   PP*1.
      M(3,3) =     PP*2.

*************************************************************************
      CALL DEBUG('ABSORB',BUG)
      IF(BUG)  THEN
        PRINT *, 'ABSORB***     TRIANG=',TRIANG
        DO 8800 I=1,3
          WRITE(*,8600) (M(I,J),J=1,3),PP
 8600     FORMAT(1X,3(F7.3,2X),5X,f7.3)
 8800   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SCAT1(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                                          SIGMAT,SIGMAS,M)
*****************************************************************************
*                                                                           *
*                            First Scatter Term                             *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,NONLOC,MNTRIA,PTNODE(MNTRIA,4),SIZE
      REAL     AREAS(MNTRIA),CORDND(SIZE,2),SIGMAT,SIGMAS

      REAL     M(3,3)

      INTEGER  I,J,CASE
      REAL     MC(3,3),MD(3,3),ALPHA,AREASQ,X(3),D,P
      LOGICAL  BUG

*****************************************************************************
      DATA (MC(1,J),J=1,3) / 4, 3, 3 /
     +     (MC(2,J),J=1,3) / 3, 6, 6 /
     +     (MC(3,J),J=1,3) / 3, 6, 6 /

      DATA (MD(1,J),J=1,3) / 4, 3, 3 /
     +     (MD(2,J),J=1,3) / 3, 1, 1 /
     +     (MD(3,J),J=1,3) / 3, 1, 1 /
*****************************************************************************

C ** Compute the coefficient of the 3rd integral.
      ALPHA = SIGMAS*(SIGMAT - SIGMAS/2.)

C ** Compute the product of the areas of the local and nonlocal elements.
      AREASQ = AREAS(TRIANG)*AREAS(NONLOC)

C ** Compute the width of the column containing these elements.
      X(1)   = CORDND(PTNODE(TRIANG,1),1)
      X(2)   = CORDND(PTNODE(TRIANG,2),1)
      D      = ABS(X(2) - X(1))

C ** Compute the constant factor in the local matrix.
      P      = -ALPHA*AREASQ/(30.*D)


C ** Determine the case.
      CALL CASEDT(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,CASE)

C -- Symmetric case, fill the local matrix using MD.
      IF(CASE .EQ. 14)  THEN
        DO 100 I=1,3
          DO 100 J=1,3
            M(I,J) = P*MC(I,J)
  100   CONTINUE
```

```
C -- Assymmetric case, fill the local matrix using M1.
      ELSE IF(CASE .EQ. 23)  THEN
        DO 110 I=1,3
          DO 110 J=1,3
            M(I,J) = P*MD(I,J)
  110   CONTINUE

      ELSE
        PRINT *, 'Impossible case in SCAT1.'

      ENDIF


*************************************************************************
      CALL DEBUG('SCAT1 ',BUG)
      IF(BUG)  THEN
        PRINT *, 'SCAT1***     TRIANG= ',TRIANG,' NONLOC=',NONLOC
        PRINT *, 'M(I,J)'
        WRITE(*,'(3(2X,F6.3))') (M(1,J),J=1,3)
        WRITE(*,'(3(2X,F6.3))') (M(2,J),J=1,3)
        WRITE(*,'(3(2X,F6.3))') (M(3,J),J=1,3)
        PRINT *, 'D             P             ALPHA             AREASQ'
        PRINT *,  D,P,ALPHA,AREASQ
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE SCAT2(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                                          SIGMAS,M)
*************************************************************************
*                                                                       *
*                          Second Scatter Term                          *
*                                                                       *
*************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,NONLOC,MNTRIA,PTNODE(MNTRIA,4),SIZE
      REAL     AREAS(MNTRIA),CORDND(SIZE,2),SIGMAS

      REAL     M(3,3)

      INTEGER  I,J,CASE
      REAL     MB(3,3),M1(3,3),AJP(3),X(3),U(3),UP(3),B(3),D,P
      LOGICAL  BUG

*************************************************************************
      DATA (MB(1,J),J=1,3) / 4, 3, 3 /
     +     (MB(2,J),J=1,3) / 3, 8, 4 /
     +     (MB(3,J),J=1,3) / 3, 4, 8 /

      DATA (M1(1,J),J=1,3) / 6, 2, 2 /
     +     (M1(2,J),J=1,3) / 2, 2, 1 /
     +     (M1(3,J),J=1,3) / 2, 1, 2 /
*************************************************************************

C ** Compute the width of the column containing these elements.
      X(1)   = CORDND(PTNODE(TRIANG,1),1)
      X(2)   = CORDND(PTNODE(TRIANG,2),1)
      D      = ABS(X(2) - X(1))

C ** Compute the mu coordinates of the local element.
      U(1)   = CORDND(PTNODE(TRIANG,1),2)
      U(2)   = CORDND(PTNODE(TRIANG,2),2)
      U(3)   = CORDND(PTNODE(TRIANG,3),2)

C ** Compute the mu-prime coordinates of the non-local element.
      UP(1)  = CORDND(PTNODE(NONLOC,1),2)
      UP(2)  = CORDND(PTNODE(NONLOC,2),2)
      UP(3)  = CORDND(PTNODE(NONLOC,3),2)

C ** Compute the x coefficient of the non-local element's linear
C --   interpolating function.
      AJP(1) = UP(2) - UP(3)
      AJP(2) = UP(3) - UP(1)
      AJP(3) = UP(1) - UP(2)

C ** Compute the constant factor in the local matrix.
      P      =  (SIGMAS/2.) * (AREAS(TRIANG)/(60.*D))
```

```
C ** Determine the case.
      CALL CASEDT(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,CASE)

C -- Symmetric case, fill the local matrix using MB.
      IF(CASE .EQ. 14)  THEN
        DO 100 I=1,3
          DO 100 J=1,3
  100       B(I) = P*MB(I,J)*U(J)
        DO 110 I=1,3
          DO 110 J=1,3
  110       M(I,J) = B(I)*AJP(J)

C -- Assymmetric case, fill the local matrix using M1.
      ELSE IF(CASE .EQ. 23)  THEN
        DO 120 I=1,3
          DO 120 J=1,3
  120       B(I) = P*M1(I,J)*U(J)
        DO 130 I=1,3
          DO 130 J=1,3
  130       M(I,J) = B(I)*AJP(J)

      ELSE
        PRINT *, 'Impossible case in SCAT2.'

      ENDIF

*****************************************************************************
      CALL DEBUG('SCAT2 ',BUG)
      IF(BUG)  THEN
        PRINT *, 'SCAT2***      TRIANG= ',TRIANG,' NONLOC=',NONLOC
        PRINT *, '  M(I,J)                      U       UP       AJP'
        WRITE(*,'(6(2X,F6.3))') (M(1,J),J=1,3),U(1),UP(1),AJP(1)
        WRITE(*,'(6(2X,F6.3))') (M(2,J),J=1,3),U(2),UP(2),AJP(2)
        WRITE(*,'(6(2X,F6.3))') (M(3,J),J=1,3),U(3),UP(3),AJP(3)
        PRINT *, 'D          P'
        PRINT *,  D,P
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE SCAT3(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,AREAS,
     +                                           SIGMAS,M)
*****************************************************************************
*                                                                           *
*                              Third Scatter Term                           *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,NONLOC,MNTRIA,PTNODE(MNTRIA,4),SIZE
      REAL     AREAS(MNTRIA),CORDND(SIZE,2),SIGMAS

      REAL     M(3,3)

      INTEGER  I,J,CASE
      REAL     MC(3,3),MD(3,3),X(3),UP(3),AJP(3),B(3),D,P
      LOGICAL  BUG

*****************************************************************************
      DATA (MC(1,J),J=1,3) / 4, 3, 3 /
     +     (MC(2,J),J=1,3) / 3, 6, 6 /
     +     (MC(3,J),J=1,3) / 3, 6, 6 /

      DATA (MD(1,J),J=1,3) / 4, 3, 3 /
     +     (MD(2,J),J=1,3) / 3, 1, 1 /
     +     (MD(3,J),J=1,3) / 3, 1, 1 /
*****************************************************************************

C ** Compute the width of the column containing these elements.
      X(1)   = CORDND(PTNODE(TRIANG,1),1)
      X(2)   = CORDND(PTNODE(TRIANG,2),1)
      D      = ABS(X(2) - X(1))

C ** Compute the mu-prime coordinates of the non-local element.
      UP(1)  = CORDND(PTNODE(NONLOC,1),2)
      UP(2)  = CORDND(PTNODE(NONLOC,2),2)
      UP(3)  = CORDND(PTNODE(NONLOC,3),2)

C ** Compute the x coefficient of the non-local element's linear
C --   interpolating function.
      AJP(1) = UP(2) - UP(3)
      AJP(2) = UP(3) - UP(1)
      AJP(3) = UP(1) - UP(2)

C ** Compute the constant factor in the local matrix.
      P      = -(SIGMAS/2.) * (AREAS(TRIANG)/(60.*D))

C ** Determine the case.
      CALL CASEDT(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,CASE)

C -- Symmetric case, fill the local matrix using MB.
      IF(CASE .EQ. 14)  THEN
```

```
        DO 100 I=1,3
          DO 100 J=1,3
  100       B(I) = P*MC(I,J)*UP(J)
        DO 110 I=1,3
          DO 110 J=1,3
  110       M(I,J) = B(I)*AJP(J)

C -- Assymmetric case, fill the local matrix using M1.
      ELSE IF(CASE .EQ. 23)  THEN
        DO 120 I=1,3
          DO 120 J=1,3
  120       B(I) = P*MD(I,J)*UP(J)
        DO 130 I=1,3
          DO 130 J=1,3
  130       M(I,J) = B(I)*AJP(J)

      ELSE
        PRINT *,'Impossible case in SCAT3.'

      ENDIF

*************************************************************************
      CALL DEBUG('SCAT3 ',BUG)
      IF(BUG)  THEN
        PRINT *, 'SCAT3***      TRIANG= ',TRIANG,' NONLOC=',NONLOC
        PRINT *, 'M(I,J)                    UP       AJP'
        WRITE(*,'(5(2X,F6.3))') (M(1,J),J=1,3),UP(1),AJP(1)
        WRITE(*,'(5(2X,F6.3))') (M(2,J),J=1,3),UP(2),AJP(2)
        WRITE(*,'(5(2X,F6.3))') (M(3,J),J=1,3),UP(3),AJP(3)
        PRINT *, 'D          P'
        PRINT *,  D,P
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE CASEDT(TRIANG,NONLOC,MNTRIA,SIZE,PTNODE,CORDND,CASE)
*****************************************************************************
*                                                                           *
*            Determine case of the local and non-local elements.            *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  TRIANG,NONLOC,MNTRIA,SIZE,PTNODE(MNTRIA,4)
      REAL     CORDND(SIZE,2)

      INTEGER  CASE

      REAL     X(3),Y(3),LOCL,NOLOCL
      LOGICAL  BUG

*****************************************************************************

C ** Compute the coordinates of the columnar ends of the triangles.
      X(1) = CORDND(PTNODE(TRIANG,1),1)
      X(2) = CORDND(PTNODE(TRIANG,2),1)
      Y(1) = CORDND(PTNODE(NONLOC,1),1)
      Y(2) = CORDND(PTNODE(NONLOC,2),1)

C ** Determine the direction in which the triangles are pointing.
      LOCL   = X(2) - X(1)
      NOLOCL = Y(2) - Y(1)

C ** Determine the case.
      IF(LOCL*NOLOCL .GT. 0.)  THEN
        CASE = 14
      ELSE
        CASE = 23
      ENDIF

*****************************************************************************
      CALL DEBUG('CASEDT',BUG)
      IF(BUG)  THEN
        PRINT *, 'CASEDT***     CASE= ',CASE
        PRINT *, 'LOCL= ',LOCL,'        NOLOCL= ',NOLOCL
      ENDIF

      RETURN
      END
```

```
      SUBROUTINE PENAL(NTRIAN,MNTRIA,MINOD,MBNOD,MLOCMT,INTNOD,BNDNOD,
     +                 PTNODE,COLUMN,PHI,PSI,MAT,M11,M12,M21,M22,PENLTY)
*****************************************************************************
*                                                                           *
*                            PENALTY FUNCTION                               *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  NTRIAN,MNTRIA,MINOD,MBNOD,MLOCMT,INTNOD,BNDNOD
      INTEGER  PTNODE(MNTRIA,4),COLUMN(32,2)
      REAL     PHI(MINOD),PSI(MBNOD)
      REAL     MAT(MLOCMT,3,3)
      REAL     M11(MINOD,MINOD),M12(MINOD,MBNOD)
      REAL     M21(MBNOD,MINOD),M22(MBNOD,MBNOD)

      REAL     PENLTY(MNTRIA)

      INTEGER  I,J,K,TRIANG,NONLOC,II,JJ,LOC
      REAL     PHILOC(3),PHIPRM(3),PEN
      LOGICAL  BUG

*****************************************************************************

      CALL DEBUG('PENAL ',BUG)
      LOC = 1
C ** Compute the cummulative penalty function element by element.
      DO 100 TRIANG=1,NTRIAN
        PEN = 0.

C ** Compute the local terms penalty
C -- Compute this element's phi vector.
        DO 110 I=1,3
          II = PTNODE(TRIANG,I)
          IF(II .LE. INTNOD)  THEN
             PHILOC(I) = PHI(II)
           ELSE
             PHILOC(I) = PSI(II-INTNOD)
           ENDIF
  110   CONTINUE

        IF(BUG)  THEN
          PRINT *, 'PENAL***      TRIANG= ',TRIANG
          PRINT *, '   PEN         PHILOC(I)  MAT(LOC,I,J)  PHILOC(J)'
        ENDIF

        DO 120 I=1,3
          DO 120 J=1,3
            PEN = PEN + PHILOC(I)*MAT(LOC,I,J)*PHILOC(J)
            IF(BUG)  THEN
              WRITE(*,6120)  PEN,PHILOC(I),I,MAT(LOC,I,J),J,PHILOC(J)
 6120         FORMAT(1X,F7.3,7X,3(F7.3,2X,I3,2X))
```

```
            ENDIF
  120    CONTINUE

C ** Compute the non-local terms penalty
         LOC = LOC + 1
         NONLOC = COLUMN(PTNODE(TRIANG,4),1)
         DO 355 K=1,COLUMN(PTNODE(TRIANG,4),2)

C -- Compute this element's non-local phi vector.
           DO 130 J=1,3
             JJ = PTNODE(NONLOC,J)
             IF(JJ .LE. INTNOD)  THEN
                PHIPRM(J) = PHI(JJ)
              ELSE
                PHIPRM(J) = PSI(JJ-INTNOD)
              ENDIF
  130      CONTINUE

         IF(BUG)  THEN
           PRINT *, 'PENAL***      TRIANG= ',TRIANG
           PRINT *, '   PEN         PHILOC(I)  MAT(LOC,I,J)  PHIPRM(J)'
         ENDIF

           DO 140 I=1,3
             DO 140 J=1,3
               PEN = PEN + PHILOC(I)*MAT(LOC,I,J)*PHIPRM(J)
               IF(BUG)  THEN
                 WRITE(*,6120)  PEN,PHILOC(I),I,MAT(LOC,I,J),J,PHIPRM(J)
               ENDIF
  140      CONTINUE

           LOC = LOC + 1
           NONLOC = NONLOC + 1
  355    CONTINUE

         PENLTY(TRIANG) = .5*PEN
  100 CONTINUE

C ** Compute the global penalty.
       PEN = 0.
       DO 200 II=1,INTNOD+BNDNOD
         DO 200 JJ=1,INTNOD+BNDNOD
           IF( (II .LE. INTNOD) .AND. (JJ .LE. INTNOD) )  THEN
             PEN = PEN + PHI(II)*M11(II,JJ)*PHI(JJ)
           ELSE IF( (II .LE. INTNOD) .AND. (JJ .GT. INTNOD) )  THEN
             PEN = PEN + PHI(II)*M12(II,JJ-INTNOD)*PSI(JJ-INTNOD)
           ELSE IF( (II .GT. INTNOD) .AND. (JJ .LE. INTNOD) )  THEN
             PEN = PEN + PSI(II-INTNOD)*M21(II-INTNOD,JJ)*PHI(JJ)
           ELSE IF( (II .GT. INTNOD) .AND. (JJ .GT. INTNOD) )  THEN
             PEN = PEN + PSI(II-INTNOD)*M22(JJ-INTNOD,II-INTNOD)*
      +                                                  PSI(JJ-INTNOD)
           ELSE
```

```
            PRINT *, 'Impossible option in PENAL:  global penalty.'
          ENDIF
  200 CONTINUE
      PEN = .5*PEN

*******************************************************************************
      IF(BUG)  THEN
        PRINT *, 'PENAL global penalty = ', PEN
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE OUTPUT(MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE,
     +                  CORDND,PHI,PSI,AREAS,PENLTY,M11,M12,M21,M22,B,
     +                  FMESH,FNODES,FELEM,FMTRXI,FMTRXB,FFORCE)
*****************************************************************************
*                                                                           *
*                           PRINT SELECTED ITEMS                            *
*                                                                           *
*****************************************************************************
      IMPLICIT UNDEFINED(A-Z)

      INTEGER  MNTRIA,MINOD,MBNOD,NTRIAN,INTNOD,BNDNOD,PTNODE(MNTRIA,4)
      REAL     AREAS(MNTRIA),PENLTY(MNTRIA),CORDND(MINOD+MBNOD,2)
      REAL     PHI(MINOD),PSI(MBNOD)
      REAL     M11(MINOD,MINOD),M12(MINOD,MBNOD)
      DOUBLE PRECISION B(MINOD)
      REAL     M21(MBNOD,MINOD),M22(MBNOD,MBNOD)
      LOGICAL  FMESH,FNODES,FELEM,FMTRXI,FMTRXB,FFORCE

      INTEGER  I,J,K,TRIANG,II
      REAL     PEN
      LOGICAL  BUG

*****************************************************************************
*       There are six groups of information.  Selected by flags.            *
*                                                                           *
*       FNODES  -  Nodal values.                                            *
*       FELEM   -  Element charactoristics.                                 *
*       FMTRXI  -  Elements of global matrix, (interior part).              *
*       FMTRXB  -  Elements of global matrix, (boundary part).              *
*       FFORCE  -  Force vector of SATE matrix equation.                    *
*       FMESH   -  Mesh definition.                                         *
*                                                                           *
*****************************************************************************

C ** Mesh definition.
      IF(FMESH)  THEN
        PRINT *, 'Mesh Definition:'
        PRINT *, '  Triangle     Global Nodes.'
        DO 50 TRIANG=1,NTRIAN
          WRITE(*,6050) TRIANG,(PTNODE(TRIANG,I),I=1,4)
 6050     FORMAT(4X,I3,8X,4(I3,3X))
   50   CONTINUE
        PRINT *
        PRINT *, '    Node      Coordinates (x,mu).'
        DO 60 I=1,INTNOD+BNDNOD
          WRITE(*,6060) I,(CORDND(I,J),J=1,2)
 6060     FORMAT(4X,I3,7X,2(F7.3,3X))
   60   CONTINUE
        PRINT *
      ENDIF

C ** Nodal values.
```

```
      IF(FNODES)  THEN
        PRINT *, 'Nodal values.'
        DO 100 I=1,INTNOD
          WRITE(*,6010) I,PHI(I)
 6010     FORMAT(2X,I3,3X,F9.4)
  100   CONTINUE
        DO 110 I=1,BNDNOD
          WRITE(*,6010) I+INTNOD,PSI(I)
  110   CONTINUE
        PRINT *
      ENDIF

C ** Element charactoristics.
      IF(FELEM)  THEN
        WRITE(*,6100)
 6100   FORMAT('Triangle Area   Penalty     Nodes(x,u)')
        DO 200 TRIANG=1,NTRIAN
          WRITE(*,6110) TRIANG,AREAS(TRIANG),PENLTY(TRIANG),
     +                  ((CORDND(PTNODE(TRIANG,J),K),K=1,2),J=1,3)
 6110     FORMAT(1X,I3,3X,F6.3,3X,F7.4,4X,3(2(F6.3,1X),3X))
          PEN = PEN + PENLTY(TRIANG)
  200   CONTINUE
        WRITE(*,6120) PEN
 6120     FORMAT(15X,F9.5)
      ENDIF

C ** Elements of global matrix, (interior part).
      IF(FMTRXI)  THEN
        PRINT *
        WRITE(*,6200)
 6200   FORMAT(' Global matrix, (interior part).')
        PRINT *
        DO 300 I=1,INTNOD
          WRITE(*,6210) (M11(I,J),J=1,INTNOD)
 6210     FORMAT(1X,'|',20(1X,F6.3))
          WRITE(*,6220)
 6220     FORMAT(1X,'|')
  300   CONTINUE
        PRINT *
      ENDIF

C ** Elements of global matrix, (boundary part).
      IF(FMTRXB)  THEN

        PRINT *
        PRINT *, ' Global matrix, (boundary part:  M12).'
        PRINT *
        DO 400 I=1,INTNOD
          WRITE(*,6210) (M12(I,J),J=1,BNDNOD)
          WRITE(*,6220)
  400   CONTINUE
```

```
        PRINT *
        PRINT *, ' Global matrix, (boundary part:  M21).'
        PRINT *
        DO 410 I=1,BNDNOD
          WRITE(*,6210) (M21(I,J),J=1,INTNOD)
          WRITE(*,6220)
  410   CONTINUE

        PRINT *
        PRINT *, ' Global matrix, (boundary part:  M22).'
        PRINT *
        DO 420 I=1,BNDNOD
          WRITE(*,6210) (M22(I,J),J=1,BNDNOD)
          WRITE(*,6220)
  420   CONTINUE
        PRINT *

      ENDIF

C ** Force vector of the SATE matrix equation.
      IF(FFORCE)  THEN
        WRITE(*,'('Force vector.')')
        DO 500 I=1,INTNOD
          WRITE(*,6500) B(I)
 6500     FORMAT(2X,F7.3)
  500   CONTINUE
        PRINT *
      ENDIF

*******************************************************************************
      CALL DEBUG('OUTPUT',BUG)
      IF(BUG)  THEN
        DO 8800 II=1,INTNOD+BNDNOD
          IF(II .LE. INTNOD)  THEN
            WRITE(*,6210) (M11(II,J),J=1,INTNOD),(M12(II,J),J=1,BNDNOD)
            WRITE(*,6220)
          ELSE
            I = II - INTNOD
            WRITE(*,6210) (M21(I,J),J=1,INTNOD), (M22(I,J),J=1,BNDNOD)
            WRITE(*,6220)
          ENDIF
 8800   CONTINUE
      ENDIF

      RETURN
      END
```

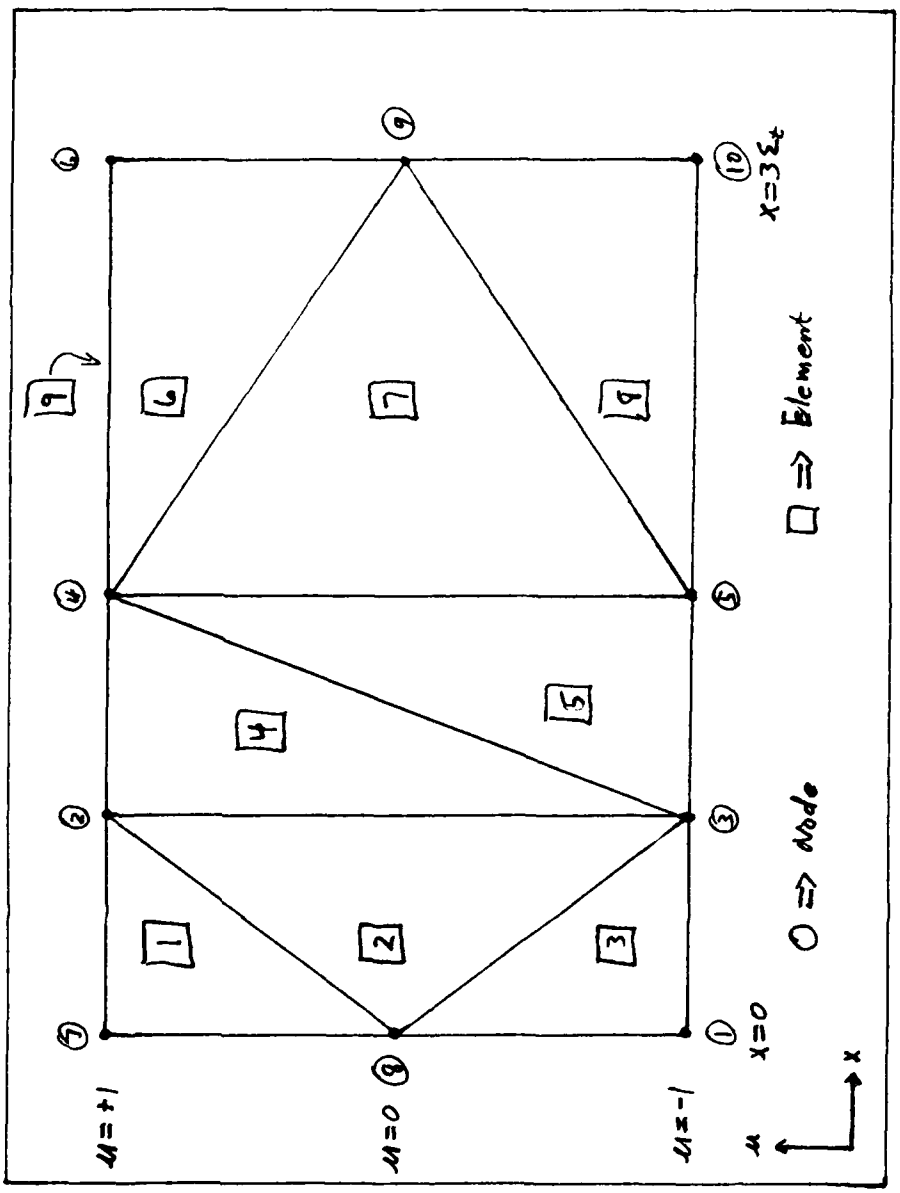
```
      SUBROUTINE  DEBUG(SUBNAM,BUG)
C ***************************************************************************
C ***                                                                     ***
C ***                      DOCUMENTATION  -  DEBUG                        ***
C ***                                                                     ***
C ***************************************************************************
C *                                                                         *
C *     Lt. Allan Goff                              August 27, 1983         *
C *-------------------------------------------------------------------------*
C *---                        DEBUGGING AID                              ---*
C *-------------------------------------------------------------------------*
C *                          ** ABSTRACT **                                 *
C *                                                                         *
C *     This routine scans a list of subroutine names to be debugged        *
C *  and if the calling routine is on the list it returns a true value      *
C *  via the variable BUG.                                                  *
C *-------------------------------------------------------------------------*
C *                        ** BACKGROUND **                                 *
C *                                                                         *
C *-------------------------------------------------------------------------*
C *                        ** FLOW-CHART **                                 *
C *                                                                         *
C *-------------------------------------------------------------------------*
C *                      ** IMPLIMENTATION **                               *
C *                                                                         *
C *     The list of subroutines to be debugged may be as large as           *
C *  100 names.  However, DEBUG quits looking thru the list when it         *
C *  finds the name QUIT88.                                                 *
C *-------------------------------------------------------------------------*
C *                   ** INTERFACE REQUIREMENTS **                          *
C *                                                                         *
C *  The parameters passed are:                                             *
C *     SUBNAM  -  The name of the calling program.                         *
C *     BUG     ^  The flag which indicates if debuggins is to be done.    *
C *-------------------------------------------------------------------------*
C *                       ** LOCAL VARIABLES **                             *
C *                                                                         *
C *     BUGEM   -  The list of subroutines to be debugged.                  *
C *     I       -  Index variable used in scanning the list.                *
C *-------------------------------------------------------------------------*
C *                        ** LIMITATIONS **                                *
C *                                                                         *
C *-------------------------------------------------------------------------*
C *                        ** REFERENCES **                                 *
C *                                                                         *
C ***************************************************************************
C ***************************************************************************
C ***                                                                     ***
C ***                         CODE  -  DEBUG                              ***
C ***                                                                     ***
C ***************************************************************************
C ************************** DECLARATIONS ***********************************
```

```
C --------- HAVE COMPILER CHECK FOR MISSPELLED VARIABLE NAMES -----------
      IMPLICIT UNDEFINED(A-Z)
C ------------------------------ INDEXES --------------------------------
      INTEGER I
C ------------------------- PASSED VARIABLES ----------------------------
      CHARACTER SUBNAM*6
      LOGICAL BUG
C --------------------------- LOCAL VARIABLES ---------------------------
      CHARACTER BUGEM(101)*6
      DATA BUGEM(1)  /'QUIT88'/
      DATA BUGEM(2)  /'SATE  '/
      DATA BUGEM(3)  /'INITAL'/
      DATA BUGEM(4)  /'ASEMBL'/
      DATA BUGEM(5)  /'STREAM'/
      DATA BUGEM(6)  /'ABSORB'/
      DATA BUGEM(7)  /'SCAT1 '/
      DATA BUGEM(8)  /'SCAT2 '/
      DATA BUGEM(9)  /'SCAT3 '/
      DATA BUGEM(10) /'CASEDT'/
      DATA BUGEM(11) /'PENAL '/
      DATA BUGEM(12) /'OUTPUT'/
      DATA BUGEM(13) /'QUIT88'/
C ****************************** EXECUTABLES ********************************
      BUG = .FALSE.
      I = 0
 2000 I = I + 1
      IF(BUGEM(I) .EQ. 'QUIT88')  GOTO 999
      IF(BUGEM(I) .EQ.  SUBNAM )  THEN
        BUG = .TRUE.
        GOTO 999
      ENDIF
      GOTO 2000
C ********************************* DONE ************************************
  999 RETURN
      END
```

Appendix C

Five Finite Element Meshes

MESH 1



MESH 1

NTRIAN	INTNOD	BNDNOD	NCOL
9	6	4	3

RANGE	SIGMAT	SIGMAS
3.	1.	.5

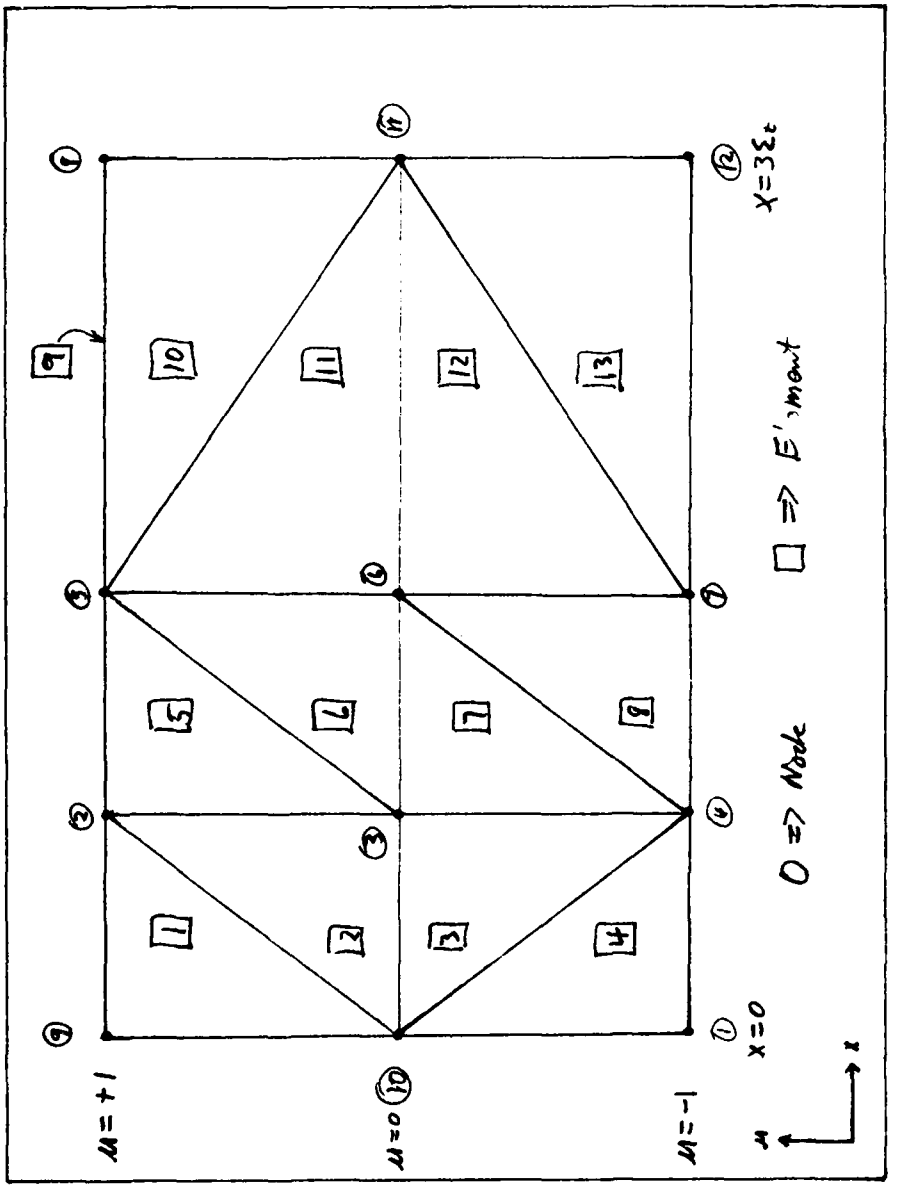
Triangle	Node1	Node2	Node3	Column
1	2	7	8	1
2	8	3	2	1
3	3	8	1	1
4	4	2	3	2
5	3	5	4	2
6	4	9	6	3
7	9	4	5	3
8	5	10	9	3
9	6	4	4	3

Column	First Element	Number of Elements
1	1	3
2	4	2
3	6	4

Node	X-axis	U-axis
1	.000	-1.000
2	.250	1.000
3	.250	-1.000
4	.500	1.000
5	.500	-1.000
6	1.000	1.000
7	.000	1.000
8	.000	.000
9	1.000	.000
10	1.000	-1.000

Node	Psi
7	1.0000
8	.0000
9	.0000
10	.0000

MESH 2



MESH 2

NTRIAN	INTNOD	BNDNOD	NCOL
13	8	4	3

RANGE	SIGMAT	SIGMAS
3.	1.	.5

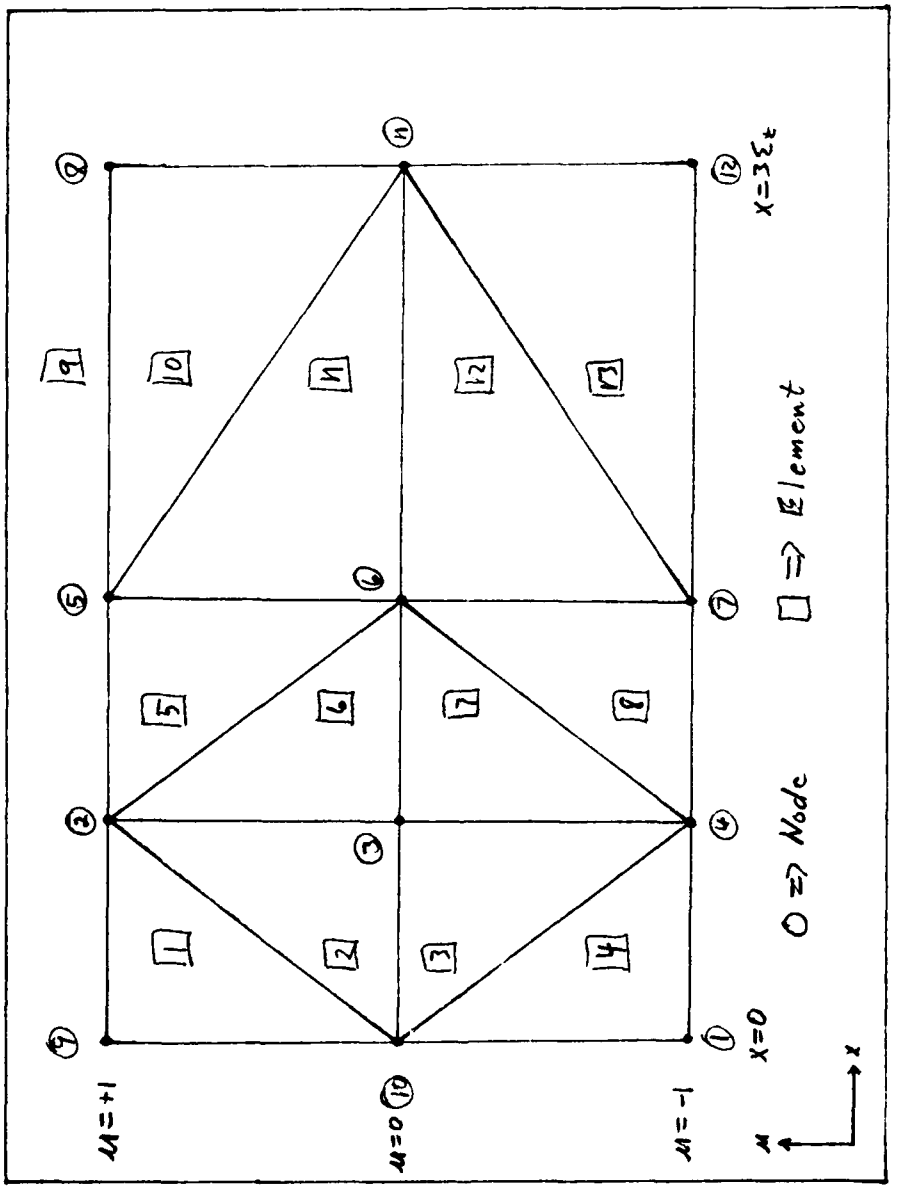
Triangle	Node1	Node2	Node3	Column
1	2	9	10	1
2	10	3	2	1
3	10	4	3	1
4	4	10	1	1
5	5	2	3	2
6	3	6	5	2
7	6	3	4	2
8	4	7	6	2
9	5	8	8	3
10	5	11	8	3
11	11	5	6	3
12	11	6	7	3
13	7	12	11	3

Column	First Element	Number of Elements
1	1	4
2	5	4
3	9	5

Node	X-axis	U-axis
1	.000	-1.000
2	.250	1.000
3	.250	.000
4	.250	-1.000
5	.500	1.000
6	.500	.000
7	.500	-1.000
8	1.000	1.000
9	.000	1.000
10	.000	.000
11	1.000	.000
12	1.000	-1.000

Node	Psi
7	1.0000
8	.0000
9	.0000
10	.0000

MESH 3



MESH 3

NTRIAN	INTNOD	BNDNOD	NCOL
13	8	4	3

RANGE	SIGMAT	SIGMAS
3.	1.	.5

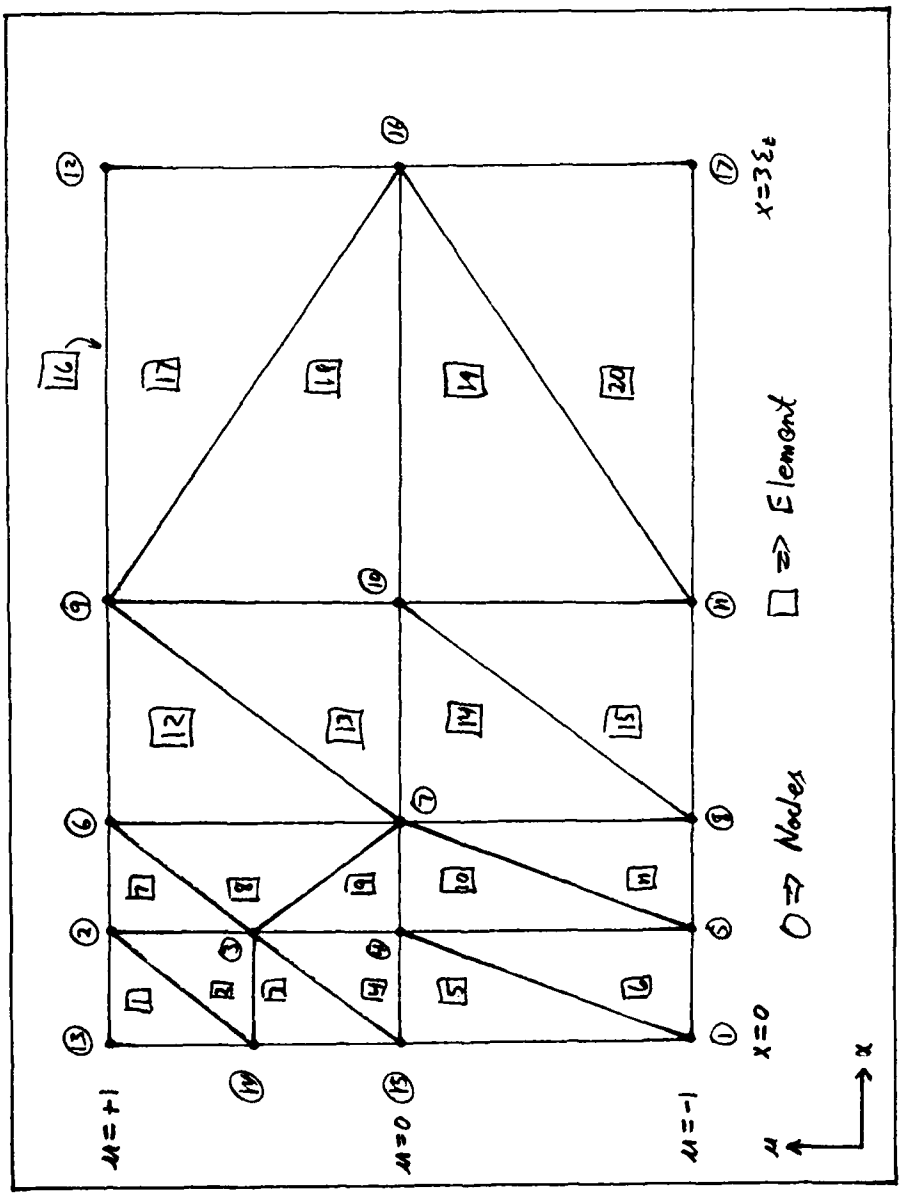
Triangle	Node1	Node2	Node3	Column
1	2	9	10	1
2	10	3	2	1
3	10	4	3	1
4	4	10	1	1
5	2	6	5	2
6	6	2	3	2
7	6	3	4	2
8	4	7	6	2
9	5	8	8	3
10	5	11	8	3
11	11	5	6	3
12	11	6	7	3
13	7	12	11	3

Column	First Element	Number of Elements
1	1	4
2	5	4
3	9	5

Node	X-axis	U-axis
1	.000	-1.000
2	.250	1.000
3	.250	.000
4	.250	-1.000
5	.500	1.000
6	.500	.000
7	.500	-1.000
8	1.000	1.000
9	.000	1.000
10	.000	.000
11	1.000	.000
12	1.000	-1.000

Node	Psi
7	1.0000
8	.0000
9	.0000
10	.0000

MESH 4



MESH 4

NTRIAN	INTNOD	BNDNOD	NCOL
20	12	5	4

RANGE	SIGMAT	SIGMAS
3.	1.	.5

Triangle	Node1	Node2	Node3	Column
1	2	13	14	1
2	14	3	2	1
3	3	14	15	1
4	15	4	3	1
5	4	15	1	1
6	1	5	4	1
7	6	2	3	2
8	3	7	6	2
9	7	3	4	2
10	7	4	5	2
11	5	8	7	2
12	9	6	7	3
13	7	10	9	3
14	10	7	8	3
15	8	11	10	3
16	9	12	12	4
17	9	16	12	4
18	16	9	10	4
19	16	10	11	4
20	11	17	16	4

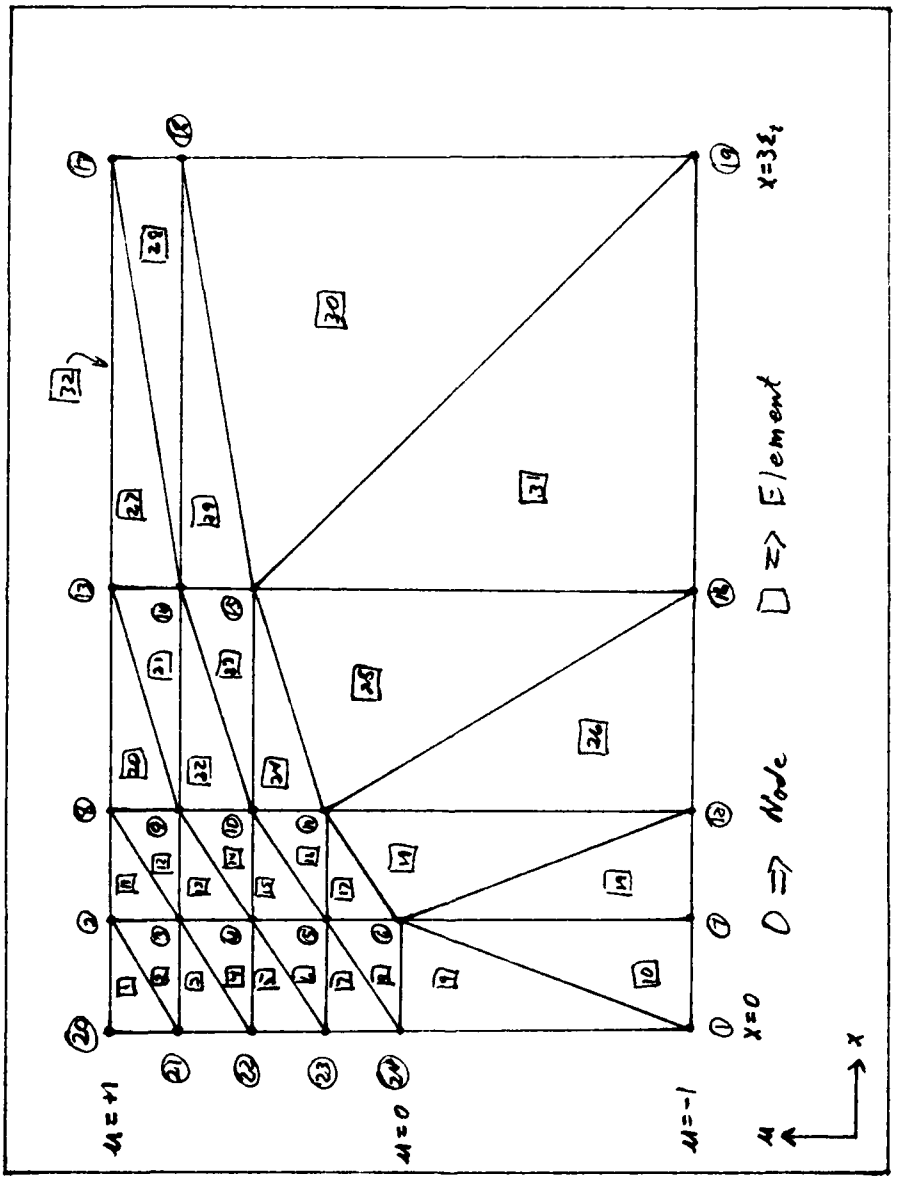
Column	First Element	Number of Elements
1	1	6
2	7	5
3	12	4
4	16	5

Node	X-axis	U-axis
1	.000	-1.000
2	.125	1.000
3	.125	.500
4	.125	.000
5	.125	-1.000
6	.250	1.000
7	.250	.000
8	.250	-1.000
9	.500	1.000
10	.500	.000
11	.500	-1.000
12	1.000	1.000
13	.000	1.000
14	.000	.500

15	.000	.000
16	1.000	.000
17	1.000	-1.000

Node	Psi
13	1.0000
14	.5000
15	.0000
16	.0000
17	.0000

MESH 5



MESH 5

NTRIAN	INTNOD	BNDNOD	NCOL
32	18	6	4

RANGE	SIGMAT	SIGMAS
3.	1.	.5

Triangle	Node1	Node2	Node3	Column
1	2	20	21	1
2	21	3	2	1
3	3	21	22	1
4	22	4	3	1
5	4	22	23	1
6	23	5	4	1
7	5	23	24	1
8	24	6	5	1
9	6	24	1	1
10	1	7	6	1
11	8	2	3	2
12	3	9	8	2
13	9	3	4	2
14	4	10	9	2
15	10	4	5	2
16	5	11	10	2
17	11	5	6	2
18	6	12	11	2
19	12	6	7	2
20	13	8	9	3
21	19	14	13	3
22	14	9	10	3
23	10	15	14	3
24	15	10	11	3
25	11	16	15	3
26	16	11	12	3
27	17	13	14	4
28	14	18	17	4
29	18	14	15	4
30	15	19	18	4
31	19	15	16	4
32	13	17	17	4

Column	First Element	Number of Elements
1	1	10
2	11	9
3	20	7
4	27	6

Node	X-axis	U-axis
1	.000	-1.000
2	.125	1.000

3	.125	.750
4	.125	.500
5	.125	.250
6	.125	.000
7	.125	-1.000
8	.250	1.000
9	.250	.750
10	.250	.500
11	.250	.250
12	.250	-1.000
13	.500	1.000
14	.500	.750
15	.500	.500
16	.500	-1.000
17	1.000	1.000
18	1.000	.750
19	1.000	-1.000
20	.000	1.000
21	.000	.750
22	.000	.500
23	.000	.250
24	.000	.000

Node	Psi
19	.0000
20	1.0000
21	.7500
22	.5000
23	.2500
24	.0000

Appendix D

Results for Local Terms

MESH 1

Nodal values.

1	-.0670
2	.3163
3	-.0878
4	.1165
5	-.0515
6	.0416
7	1.0000
8	.0000
9	.0000
10	.0000

Triangle	Area	Penalty
1	.375	.2443
2	.750	.0129
3	.375	.0013
4	.750	.0328
5	.750	.0021
6	.750	.0035
7	1.500	.0027
8	.750	.0008
9	.000	.0000
		.30029

MESH 2

Nodal values.	
1	.0087
2	.4131
3	-.0548
4	.0115
5	.1566
6	-.0323
7	.0080
8	.0559
9	1.0000
10	.0000
11	.0000
12	.0000

Triangle	Area	Penalty
1	.375	.2138
2	.375	.0098
3	.375	.0005
4	.375	.0000
5	.375	.0364
6	.375	.0011
7	.375	.0004
8	.375	.0000
9	.000	.0000
10	.750	.0062
11	.750	.0026
12	.750	.0002
13	.750	.0000
		.27105

MESH 3

Nodal values.

1	.0118
2	.3894
3	-.0501
4	.0155
5	.1589
6	-.0593
7	.0126
8	.0567
9	1.0000
10	.0000
11	.0000
12	.0000

Triangle	Area	Penalty
1	.375	.2206
2	.375	.0087
3	.375	.0004
4	.375	.0000
5	.375	.0308
6	.375	.0074
7	.375	.0005
8	.375	.0002
9	.000	.0000
10	.750	.0064
11	.750	.0026
12	.750	.0006
13	.750	.0000
		.27822

MESH 4

Nodal values.

1	.0104
2	.6367
3	.2674
4	-.0195
5	.0107
6	.3850
7	-.0325
8	.0105
9	.1441
10	-.0260
11	.0069
12	.0515
13	1.0000
14	.5000
15	.0000
16	.0000
17	.0000

Triangle	Area	Penalty
1	.094	.1109
2	.094	.0376
3	.094	.0116
4	.094	.0011
5	.188	.0001
6	.188	.0000
7	.094	.0478
8	.188	.0127
9	.094	.0009
10	.188	.0001
11	.188	.0000
12	.375	.0324
13	.375	.0009
14	.375	.0001
15	.375	.0000
16	.000	.0000
17	.750	.0053
18	.750	.0023
19	.750	.0001
20	.750	.0000
		.26395

MESH 5

Nodal	values.
1	-.0008
2	.6569
3	.4936
4	.2281
5	.0540
6	.0001
7	-.0009
8	.4228
9	.3573
10	.0965
11	.0027
12	-.0011
13	.1313
14	.1036
15	.0051
16	-.0015
17	.0281
18	.0135
19	.0000
20	1.0000
21	.7500
22	.5000
23	.2500
24	.0000

Triangle	Area	Penalty
1	.047	.0635
2	.047	.0342
3	.047	.0258
4	.047	.0164
5	.047	.0095
6	.047	.0030
7	.047	.0010
8	.047	.0000
9	.188	.0000
10	.188	.0000
11	.047	.0284
12	.047	.0128
13	.047	.0090
14	.047	.0045
15	.047	.0018
16	.047	.0002
17	.047	.0001
18	.234	.0000
19	.188	.0000
20	.094	.0210
21	.188	.0015
22	.094	.0084

23	.094	.0010
24	.094	.0004
25	.563	.0000
26	.469	.0000
27	.188	.0023
28	.188	.0010
29	.188	.0007
30	1.313	.0001
31	1.125	.0000
32	.000	.0000
		.24672

Appendix E

Results for Non-Local Terms

MESH 1

Nodal values.	
1	.0585
2	.4027
3	.0281
4	.1990
5	.0226
6	.0969
7	1.0000
8	.0000
9	.0000
10	.0000

Triangle	Area	Penalty
1	.375	.0918
2	.750	.0098
3	.375	-.0004
4	.750	.0167
5	.750	.0002
6	.750	.0037
7	1.500	.0027
8	.750	.0000
9	.000	.0000
		.12452

Nodal values.

1	.1403
2	.4991
3	.0223
4	.1332
5	.2464
6	.0379
7	.0912
8	.1257
9	1.0000
10	.0000
11	.0000
12	.0000

Triangle	Area	Penalty
1	.375	.0790
2	.375	.0037
3	.375	-.0004
4	.375	-.0007
5	.375	.0188
6	.375	.0008
7	.375	.0000
8	.375	.0003
9	.000	.0000
10	.750	.0053
11	.750	.0021
12	.750	-.0002
13	.750	.0007
		.10942

MESH 3

Nodal values.	
1	.1328
2	.4701
3	.0292
4	.1256
5	.2357
6	.0082
7	.0843
8	.1185
9	1.0000
10	.0000
11	.0000
12	.0000

Triangle	Area	Penalty
1	.375	.0828
2	.375	.0033
3	.375	-.0003
4	.375	-.0006
5	.375	.0169
6	.375	.0043
7	.375	-.0003
8	.375	.0006
9	.000	.0000
10	.750	.0050
11	.750	.0019
12	.750	-.0002
13	.750	.0006
		.11401

MESH 4

Nodal values.

1	.1806
2	.6988
3	.3479
4	.0758
5	.1699
6	.4751
7	.0797
8	.1466
9	.2389
10	.0575
11	.0971
12	.1239
13	1.0000
14	.5000
15	.0000
16	.0000
17	.0000

Triangle	Area	Penalty
1	.094	.0386
2	.094	.0119
3	.094	.0024
4	.094	-.0001
5	.188	.0003
6	.188	-.0013
7	.094	.0188
8	.188	.0060
9	.094	.0001
10	.188	-.0003
11	.188	-.0002
12	.375	.0164
13	.375	.0010
14	.375	.0000
15	.375	.0004
16	.000	.0000
17	.750	.0048
18	.750	.0020
19	.750	-.0001
20	.750	.0008
		.10168

MESH 5

Nodal values.	
1	.1410
2	.7067
3	.5450
4	.3049
5	.1803
6	.0953
7	.1282
8	.4924
9	.4275
10	.1916
11	.1102
12	.1033
13	.2067
14	.1693
15	.0788
16	.0606
17	.0809
18	.0613
19	.0000
20	1.0000
21	.7500
22	.5000
23	.2500
24	.0000

Triangle	Area	Penalty
1	.047	.0234
2	.047	.0127
3	.047	.0086
4	.047	.0051
5	.047	.0025
6	.047	.0004
7	.047	.0000
8	.047	-.0002
9	.188	.0005
10	.188	-.0012
11	.047	.0117
12	.047	.0057
13	.047	.0035
14	.047	.0020
15	.047	.0007
16	.047	.0002
17	.047	.0000
18	.234	-.0007
19	.188	.0001
20	.094	.0103
21	.188	.0009
22	.094	.0044

23	.094	.0009
24	.094	.0003
25	.563	.0001
26	.469	.0010
27	.188	.0019
28	.188	.0010
29	.188	.0007
30	1.313	.0004
31	1.125	.0004
32	.000	.0000
		.09727

Appendix F

Sample Output of Debug Option

```
STREAM***       TRIANG= 1
    .333     -.333       .000          1.000         .577         3.000
.577
   -.333      .333       .000          1.000        -.577         3.000
.577
    .000      .000       .000           .000         .000         3.000
.577
ASEMBL***
ABSORB***       TRIANG= 1
    .063      .031      .031           .031
    .031      .063      .031           .031
    .031      .031      .063           .031
ASEMBL***
SCAT1***        TRIANG=  1  NONLOC= 1
M(I,J)
   -.009    -.007    -.007
   -.007    -.014    -.014
   -.007    -.014    -.014
D              P              ALPHA              AREASQ
 .750000000e+00 -.234375009e-02  .375000000e+00  .140625000e+00
ASEMBL***
SCAT2***        TRIANG=  1  NONLOC= 1
  M(I,J)                      U        UP       AJP
   -.009    -.007    -.007    1.000    1.000    1.000
   -.007    -.014    -.014    1.000    1.000   -1.000
   -.007    -.014    -.014     .000     .000     .000
D          P
 .750000000e+00  .208333344e-02
ASEMBL***
SCAT3***        TRIANG=  1  NONLOC= 1
M(I,J)                      UP       AJP
   -.009    -.007    -.007    1.000    1.000
   -.007    -.014    -.014    1.000   -1.000
   -.007    -.014    -.014     .000     .000
D          P
 .750000000e+00 -.208333344e-02
ASEMBL***
SCAT1***        TRIANG=  1  NONLOC= 2
M(I,J)
   -.019    -.014    -.014
   -.014    -.005    -.005
   -.014    -.005    -.005
D              P              ALPHA              AREASQ
 .750000000e+00 -.468750019e-02  .375000000e+00  .281250000e+00
ASEMBL***
SCAT2***        TRIANG=  1  NONLOC= 2
  M(I,J)                      U        UP       AJP
    .000     .000     .000    1.000     .000   -2.000
    .000     .000     .000    1.000   -1.000    1.000
    .000     .000     .000     .000    1.000    1.000
D          P
 .750000000e+00  .208333344e-02
```

```
ASEMBL***
SCAT3***      TRIANG=  1  NONLOC= 2
M(I,J)                          UP          AJP
   .013     .013     .013      .000    -2.000
  -.002    -.002    -.002    -1.000     1.000
  -.002    -.002    -.002     1.000     1.000
D           P
 .750000000e+00 -.208333344e-02
ASEMBL***
SCAT1***      TRIANG=  1  NONLOC= 3
M(I,J)
  -.009    -.007    -.007
  -.007    -.014    -.014
  -.007    -.014    -.014
D               P                ALPHA                AREASQ
 .750000000e+00 -.234375009e-02   .375000000e+00   .140625000e+00
ASEMBL***
SCAT2***      TRIANG=  1  NONLOC= 3
  M(I,J)                          U          UP         AJP
  -.009    -.007    -.007     1.000    -1.000     1.000
  -.007    -.014    -.014     1.000      .000      .000
  -.007    -.014    -.014      .000    -1.000    -1.000
D           P
 .750000000e+00  .208333344e-02
ASEMBL***
SCAT3***      TRIANG=  1  NONLOC= 3
M(I,J)                          UP          AJP
  -.009    -.007    -.007    -1.000     1.000
  -.007    -.014    -.014      .000      .000
  -.007    -.014    -.014    -1.000    -1.000
D           P
 .750000000e+00 -.208333344e-02
ASEMBL***

Global matrix, (interior part).

|    .00    .00    .00    .00    .00    .00
|
|    .00    .33    .00    .00    .00    .00
|
|    .00    .00    .00    .00    .00    .00
|
|    .00    .00    .00    .00    .00    .00
|
|    .00    .00    .00    .00    .00    .00
|
|    .00    .00    .00    .00    .00    .00
|

 Global matrix, (boundary part:  M12).
```

```
|   .00    .00    .00    .00
|
|  -.35   -.01    .00    .00
|
|   .00    .00    .00    .00
|
|   .00    .00    .00    .00
|
|   .00    .00    .00    .00
|
|   .00    .00    .00    .00
|
```

Global matrix, (boundary part: M21).

```
|   .00   -.36    .00    .00    .00    .00
|
|   .00   -.03    .00    .00    .00    .00
|
|   .00    .00    .00    .00    .00    .00
|
|   .00    .00    .00    .00    .00    .00
|
```

Global matrix, (boundary part: M22).

```
|   .30   -.06    .00    .00
|
|  -.06   -.03    .00    .00
|
|   .00    .00    .00    .00
|
|   .00    .00    .00    .00
|
```

Appendix G

First Scatter Term - Antisymmetric Cases

In this appendix, the local matrix for the first scatter term, for the antisymmetric cases (Cases 2 and 3), is derived.

The starting point for this derivation is the extremization integral.

$$I_3 = -\tfrac{1}{2}\alpha \int \phi \int_{\mu'} \phi' \, d\mu' \, dA \qquad (301)$$

Starting with the usual finite element approximation, the flux at the local element is

$$\phi = \sum_i^3 \phi_i N_i = \sum_i^3 \phi_i \frac{1}{2A}(a_i x + b_i \mu + c_i) \qquad (302)$$

regrouping terms,

$$\phi = \sum_i^3 \frac{\phi_i}{2A} a_i x + \sum_i^3 \frac{\phi_i}{2A} b_i \mu + \sum_i^3 \frac{\phi_i}{2A} c_i \qquad (303)$$

and defining

$$\alpha_1 = \sum_i^3 \frac{\phi_i}{2A} a_i \qquad (304)$$

$$\alpha_2 = \sum_i^3 \frac{\phi_i}{2A} b_i \qquad (305)$$

$$\alpha_3 = \sum_i^3 \frac{\phi_i}{2A} \qquad (306)$$

the finite element approximation becomes

$$\phi = \alpha_1 x + \alpha_2 \mu + \alpha_3 \qquad (307)$$

Similarly, the finite element approximation for the flux at the non-local element is

$$\phi' = \beta_1 x + \beta_2 \mu' + \beta_3 \tag{308}$$

where the coefficients are defined as

$$\beta_1 = \sum_j^3 \frac{d_j'}{2A'} a_j' \tag{309}$$

$$\beta_2 = \sum_j^3 \frac{d_j'}{2A'} b_j' \tag{310}$$

$$\beta_3 = \sum_j^3 \frac{d_j'}{2A'} c_j' \tag{311}$$

Thus, the first scattering term extremization integral becomes

$$I_3 = -\tfrac{1}{2}\alpha \int_{x_p}^{x_q} \left\{ \int_{\underline{\mu}}^{\bar{\mu}} \left[a_1 x + a_2 \mu + a_3 \right] d\mu \int_{\underline{\mu}'}^{\bar{\mu}'} \left[\beta_1 x + \beta_2 \mu' + \beta_3 \right] d\mu' \right\} dx \tag{312}$$

where $\bar{\mu}$ and $\underline{\mu}$ are, respectively, the upper and lower edges of the local element, $\bar{\mu}'$ and $\underline{\mu}'$ are, respectively, the upper and lower edges of the non-local element, and $x_p - x_q = \alpha$. Integrating over μ and μ'

$$I_3 = -\tfrac{1}{2}\alpha \int_{x_p}^{x_q} \left\{ \left[(a_1 x + a_3)\mu + \tfrac{1}{2} a_2 \mu^2 \right] \Big|_{\underline{\mu}}^{\bar{\mu}} \right.$$
$$\left. \left[(\beta_1 x + \beta_3)\mu' + \tfrac{1}{2}\beta_2 \mu'^2 \right] \Big|_{\underline{\mu}'}^{\bar{\mu}'} \right\} dx \tag{313}$$

or

$$I_3 = -\tfrac{1}{2}\alpha \int_{x_p}^{x_s} \Big\{ \Big[(\alpha_1 x + \alpha_3)(\bar{u} - \underline{u}) + \tfrac{1}{2}\alpha_2 (\bar{u}^2 - \underline{u}^2) \Big] \Big[(\beta_1 x + \beta_3)(\bar{u}' - \underline{u}') + \tfrac{1}{2}\beta_2 (\bar{u}'^2 - \underline{u}'^2) \Big] \Big\} dx \tag{314}$$

The quadratic terms involving u and u' can be simplified

$$\bar{u}^2 - \underline{u}^2 = (\bar{u} - \underline{u})(\bar{u} + \underline{u}) \tag{315}$$

$$\bar{u}'^2 - \underline{u}'^2 = (\bar{u}' - \underline{u}')(\bar{u}' + \underline{u}') \tag{316}$$

The difference terms are

$$\bar{u} - \underline{u} = \lambda_{12} \frac{2A}{d^2} (x - x_1) \tag{317}$$

$$\bar{u} - \underline{u}' = \lambda_{13} \frac{2A'}{d^2} (x - y_1) \tag{318}$$

The sum terms are

$$\bar{u} + \underline{u} = \frac{1}{x_2 - x_1} [\ell x - k y_1] \tag{319}$$

$$\bar{u}' + \underline{u}' = \frac{1}{y_2 - y_1} [\ell' x - k' y_1] \tag{320}$$

where the ℓ's and k's are now

$$\ell = \mu_3 + \mu_2 - 2\mu_1 \qquad k = \mu_3 + \mu_2 - 2\mu_1 \frac{x_2}{x_1} \tag{321}$$

$$\ell' = \mu_3' + \mu_2' - 2\mu_1' \qquad k' = \mu_3' + \mu_2' - 2\mu_1' \frac{y_2}{y_1} \tag{322}$$

Thus,

$$I_3 \doteq -\tfrac{1}{2}\alpha \pi_{14} \frac{4AA'}{d^4} \int_{x_p}^{x_q} \Big\{ (x-\alpha_1)\Big[\alpha_1 x + \alpha_3 + \frac{\alpha_2}{2(x_2-x_1)}(\ell x - k y_1)\Big]$$

$$(x-y_1)\Big[\beta_1 x + \beta_3 + \frac{\beta_2}{2(y_2-y_1)}(\ell' x - k' y_1)\Big]\Big\} dx \tag{323}$$

Later algebraic manipulations can be reduced by a change of variables. Defining kappa as

$$\chi = x - x_1 \tag{324}$$

Thus,

$$d\chi = dx \tag{325}$$

and the new integration limits are

$$\chi_p = x_p - x_1 \qquad \chi_q = x_q - x_1 \tag{326}$$

Making the kappa substitution into the first bracketed term, it becomes

$$\Big[\alpha_1 \chi + \alpha_1 x_1 + \alpha_3 + \frac{\alpha_2}{2(x_2-x_1)}\big(\ell\chi + (\ell-k)x_1\big)\Big] \tag{327}$$

$$\left[\alpha_1 K + \frac{\alpha_2}{2(x_2-x_1)} \ell K + \alpha_1 x_1 + \frac{\alpha_2}{2(x_2-x_1)} (\mu_3 + \mu_2 - 2\mu_1 - \mu_3 - \mu_2 + 2\mu_1 \frac{x_2}{x_1}) x_1 + \alpha_3\right] \quad (328)$$

$$\left[(\alpha_1(x_2-x_1) + \tfrac{1}{2}\alpha_2 \ell) \frac{K}{x_2-x_1} + \alpha_1 x_1 + \alpha_2 \mu_1 + \alpha_3\right] \quad (329)$$

let

$$C_1 = \alpha_1 (x_2 - x_1) + \tfrac{1}{2}\alpha_2 \ell \quad (330)$$

and

$$C_2 = \alpha_1 x_1 + \alpha_2 \mu_1 + \alpha_3 \quad (331)$$

Making the kappa substitution into the second bracketed term, it becomes

$$\left[\beta_1 K + \beta_1 x_1 + \beta_3 + \frac{\beta_2}{2(y_2-y_1)} (\ell' K + \ell' x_1 - k' y_1)\right] \quad (332)$$

For the antisymmetric cases, $y_1 = x_2$. Thus, the parenthesised term becomes

$$\left(\ell' K + (\mu_3' + \mu_2' - 2\mu_1') x_1 - (\mu_3' + \mu_2' - 2\mu_1' \frac{x_1}{x_2}) x_2\right) \quad (333)$$

or

$$\left(\ell' K + (\mu_3' + \mu_2')(x_1 - x_2)\right) \quad (334)$$

Thus,

$$\left[(\beta_1 (y_2 - y_1) + \tfrac{1}{2}\beta_2 \ell') \frac{K}{y_2-y_1} + \beta_1 x_1 + \tfrac{1}{2}\beta_2 (\mu_3' + \mu_3') + \beta_3\right] \quad (335)$$

Let

$$C_3 = \beta_1 (y_2 - y_1) + \frac{1}{2} \beta_2 \ell' \tag{336}$$

and

$$C_4 = \beta_1 x_1 + \frac{1}{2} \beta_2 (\mu_2' + \mu_3') + \beta_3 \tag{337}$$

Expressing the integral in terms of these constraints yields

$$I_3 = -\tfrac{1}{2} \alpha \frac{4AA'}{d^4} \int_{x_p}^{x_s} \left\{ x \left[C_1 \frac{x}{x_2 - x_1} + C_2 \right] \right.$$

$$\left. (x + x_1 - x_2) \left[C_3 \frac{x}{x_1 - x_2} + C_4 \right] \right\} dx \tag{338}$$

Collecting terms

$$I_3 = -\tfrac{1}{2} \alpha \frac{4AA'}{d^4} \int_{x_p}^{x_s} \left\{ \left[C_1 \frac{x^2}{x_2 - x_1} + C_2 x \right] \right.$$

$$\left. \left[-C_3 \frac{x^2}{x_2 - x_1} + (C_3 + C_4) x - C_4 (x_1 - x_1) \right] \right\} dx \tag{339}$$

and

$$I_3 = -\tfrac{1}{2} \alpha \frac{4AA'}{d^4} \int_{x_p}^{x_s} \left\{ -C_1 C_3 \frac{x^4}{(x_2 - x_1)^2} + (C_1 C_3 + C_1 C_4 - C_2 C_3) \frac{x^3}{x_2 - x_1} \right.$$

$$\left. + (C_2 C_3 + C_2 C_4 - C_1 C_4) x^2 - C_2 C_4 (x_2 - x_1) x \right\} dx \tag{340}$$

Integrating

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{d^4}\left\{ -\frac{c_1c_3}{5}\frac{x^5}{d^2} + \frac{1}{4}(c_1c_3 + c_1c_4 - c_2c_3)\frac{x^4}{x_2-x_1} \right.$$

$$\left. + \frac{1}{3}(c_2c_3 + c_2c_4 - c_1c_4)x^3 - \frac{c_2c_4}{2}(x_2-x_1)x^2 \right\}\Bigg|_{x_1}^{x_2} \qquad (341)$$

The four constants can be expressed in terms of the nodal values of the fluxes. The first constant is

$$c_1 = a_1x_2 - a_1x_1 + \tfrac{1}{2}a_2(u_3 + u_2 - 2u_1) \qquad (342)$$

or, utilizing the equivalence between x_2 and x_3

$$\begin{aligned} c_1 = &-a_1x_1 - a_2u_1 - a_3 \\ &+ \tfrac{1}{2}(a_1x_2 + a_2u_2 + a_3) \\ &+ \tfrac{1}{2}(a_1x_3 + a_2u_3 + a_3) \end{aligned} \qquad (343)$$

Thus, from the finite element approximation, it becomes

$$c_1 = -\phi_1 + \tfrac{1}{2}(\phi_2 + \phi_3) \qquad (344)$$

The second constant becomes

$$c_2 = a_1x_1 + a_2u_1 + a_3 = \phi_1 \qquad (345)$$

The third constant is

$$c_3 = \beta_1y_2 - \beta_1y_1 + \tfrac{1}{2}\beta_2(u_3' + u_2' - 2u_1') \qquad (346)$$

or, utilizing the equivalence between y_2 and y_3

$$\zeta_3 = -\beta_1 y_1 - \beta_2 u_1' - \beta_3 + \frac{1}{2}(\beta_1 y_2 + \beta_2 u_2' + \beta_3) + \frac{1}{2}(\beta_1 y_3 + \beta_2 u_3' + \beta_3) \tag{347}$$

Thus, from the finite element approximation, it becomes

$$\zeta_3 = -\phi_1' + \frac{1}{2}(\phi_2' + \phi_3') \tag{348}$$

The fourth constant becomes

$$\zeta_4 = \beta_1 y_2 + \frac{1}{2}\beta_2(u_2' + u_3') + \beta_3 = \frac{1}{2}(\phi_2' + \phi_3') \tag{349}$$

For Case 2

$$X_p = X_2 - X_1 = -d \tag{350}$$

and

$$X_q = X_1 - X_1 = 0 \tag{351}$$

Thus, the integral becomes

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{60d^4}\left\{-12\zeta_1\zeta_3\frac{X^5}{d^2} - 15(\zeta_1\zeta_3 + \zeta_1\zeta_4 - \zeta_2\zeta_3)\frac{X^4}{d} + 20(\zeta_2\zeta_3 + \zeta_2\zeta_4 - \zeta_1\zeta_4)X^3 + 30\zeta_2\zeta_4 d X^2\right\}\Big|_{-d}^{0} \tag{352}$$

or

$$I_3 = -\tfrac{1}{2}\alpha \frac{-4AA'}{60d}\left\{12C_1C_3 - 15(C_1C_3 + C_2C_4 - C_2C_3)\right.$$

$$\left. -20(C_2C_3 + C_2C_4 - C_1C_4) + 30\,C_2C_4\right\} \tag{353}$$

Collecting terms, yields

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{60d}\left\{3C_1C_3 - 5C_1C_4 + 5C_2C_3 - 10C_2C_4\right\} \tag{354}$$

For Case 3

$$X_p = X_1 - X_1 = 0 \tag{355}$$

and

$$X_q = X_2 - X_1 = d \tag{356}$$

Thus, the integral becomes

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{60d^4}\left\{-12C_1C_3\frac{X^5}{d^5} + 15(C_1C_3 + C_1C_4 - C_2C_3)\frac{X^4}{d}\right.$$

$$\left. +20(C_2C_3 + C_2C_4 - C_1C_4)X^3 - 30\,C_2C_4\,d\,X^2\right\}\Big|_0^d \tag{357}$$

or

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{60d}\left\{-12C_1C_3 + 15(C_1C_3 + C_1C_4 - C_2C_3)\right.$$

$$\left. +20(C_2C_3 + C_2C_4 - C_1C_4) - 30C_2C_4\right\} \tag{358}$$

Collecting terms, yields

$$I_3 = -\tfrac{1}{2}\alpha \frac{4AA'}{60d}\{3C_1C_3 - 5C_1C_4 + 5C_2C_3 - 10C_2C_4\} \qquad (359)$$

which is the same as for Case 2.

Substituting for the constants

$$\begin{aligned} I_3 = -\tfrac{1}{2}\alpha \frac{AA'}{60d}\{ & 3(-2\phi_1 + \phi_2 + \phi_3)(-2\phi_1' + \phi_2' + \phi_3') \\ & -5(-2\phi_1 + \phi_2 + \phi_3)(\phi_2' + \phi_3') \\ & +5(2\phi_1)(-2\phi_1' + \phi_2' + \phi_3') \\ & -10(2\phi_1)(\phi_2' + \phi_3')\} \end{aligned} \qquad (360)$$

Rearranging terms

$$\begin{aligned} I_3 = -\tfrac{1}{2}\alpha \frac{AA'}{60d}\{ & 3(-2\phi_1 + \phi_2 + \phi_3)(-2\phi_1' + \phi_2' + \phi_3') \\ & -5(2\phi_1)(2\phi_1') - 5(\phi_2 + \phi_3)(\phi_2' + \phi_3') \\ & +5(2\phi_1)(\phi_2' + \phi_3') + 5(2\phi_1)(\phi_2' + \phi_3') \\ & +10(2\phi_1)(\phi_2' + \phi_3')\} \end{aligned} \qquad (361)$$

Collecting terms

$$I_3 = -\tfrac{1}{2}\alpha \frac{AA'}{60d}\{-8\phi_1\phi_1' - 6\phi_1(\phi_2' + \phi_3') - 6\phi_1'(\phi_2+\phi_3) - 2(\phi_2+\phi_3)(\phi_2'+\phi_3')\} \quad (362)$$

in matrix form

$$I_3 = \tfrac{1}{2}\alpha \frac{AA'}{30d}[\phi_1 \ \phi_2 \ \phi_3]\begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix}\begin{bmatrix} \phi_1' \\ \phi_2' \\ \phi_3' \end{bmatrix} \quad (363)$$

Therefore, the local matrix for the third scatter term, for the antisymmetric cases, is

$$M_{ij}^{(3)} = \alpha \frac{AA'}{30d}\begin{bmatrix} 4 & 3 & 3 \\ 3 & 1 & 1 \\ 3 & 1 & 1 \end{bmatrix}. \quad (364)$$

VITA

First Lieutenant Allan D. Goff was born on 27 November 1955 in Waco, Texas. After five years, his family moved to Astoria, Oregon, where he spent the remainder of his childhood. He graduated from Astoria High School in June of 1974 and continued on to get his B.S. in physics from Oregon State University. In 1980, Goff joined the United States Air Force and was assigned as project engineer at Space Division at Los Angeles AFS. While assigned there, he was active in the development of the Air Force's Anti-Satellite weapons system. In August, 1982, Goff was reassigned to the Air Force Institute of Technology at Wright-Patterson AFB. While assigned there, he pursued a Master of Science degree in Nuclear Engineering. Upon graduation, he will be reassigned to AFTAC at McClellan AFB in Sacramento, California. A. D. Goff is married to the former Jamie Lynn Robertson of Rolling Hills Estates, California.

Permanent Address: 3393 Harrison Ave.
Astoria, OR 97103

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): ~~AFIT/EN/GNE84~~ AFIT/GNE/PH/84M-4	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/GNE	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, OHio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): Allan D. Goff, B.S., 1st Lt, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____; 14. DATE OF REPORT (Yr., Mo., Day): 1984 March; 15. PAGE COUNT: 173	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	
18	11		Finite element analysis; Neutron transport theory; Numerical analysis; Transport properties; Variational Methods; Variational principles

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: FINITE ELEMENT SOLUTION OF A SELF-ADJOINT TRANSPORT EQUATION IN ONE DIMENSION

Thesis Chairman: Dr. Donn Shankland

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433
7 May 84

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Donn Shankland	22b. TELEPHONE NUMBER (Include Area Code): (513) 255-4498	22c. OFFICE SYMBOL: AFIT/GNE

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

A self-adjoint form of the transport equation was derived, and expressed as an extremization integral. The finite element equations were derived from the extremization integral for the one-dimensional time independent homogeneous transport equation with isotropic scatter. These equations were implemented in FORTRAN on a VAX 11/780 and used to solve a simple benchmark problem.

The finite element solution, for a small mesh of 32 elements, was compared to results from a numerical technique known as Ln. The solutions differed by about 35 percent. Larger meshes were not run because an automatic mesh refinement routine was not available. The large difference between the Ln solution and the finite difference solution is attributed to residual errors in the coding of the finite element equations for the scatter terms.